Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Recherche et stockage d'information textuelle

Michalski, Jean

Award date:
1992

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 23. Jun. 2020

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Recherche et stockage d'information textuelle

par Jean Michalski

Promoteur : Professeur J. Fichefet

Mémoire présenté
en vue de l'obtention du titre
de Licencié et Maître en Informatique

Année académique 1991/1992

Résumé

Les documents textuels présentent des problèmes particuliers lorsqu'il s'agit de rechercher ou de stocker l'information. La première partie du mémoire identifie les concepts théoriques à la base des solutions, et la deuxième partie montre, à travers une petite expérience menée sur un livre d'homéopathie, les avantages et inconvénients de deux approches, celle de Folio Views et celle de Spirit.

Abstract

This essay's aim is to show where the problems bound to text storage and retrieval lie. This is achieved in two steps : in the first part, a theoretical overview of the field is given, while the second part is devoted to a closer look at two products, Folio Views and Spirit. These are compared thanks to a small experiment designed for homeopathic material.

Remerciements

D'abord, je tiens à remercier M. Fichefet, chez qui j'ai toujours trouvé une oreille attentive et un avis constructif, malgré mon attitude parfois "déconcertante". Mme de Baenst m'a aussi beaucoup aidé, non seulement par sa disponibilité (et celle de son ordinateur !), mais aussi et surtout par ses conseils, ses encouragements et son sourire. Je n'oublierai pas M. Orban de Xivry, sa gentillesse et l'attention qu'il a portée à ce travail. Le docteur Schroyens, quant à lui, a aimablement accepté de jouer le rôle de l'utilisateur.

Merci à ma famille, qui m'a donné cette seconde chance (je pense surtout à mon père, qui a été particulièrement compréhensif en ces circonstances), et qui, dans la vie de tous les jours, a permis la réalisation de ce mémoire. Merci à l'abbé Jacques Jordant, pour son support tout au long de ces moments parfois difficiles pour lui. Xavier Decornet et Mme Mazur m'ont procuré une aide précieuse, comme tant d'autres que je n'ai pas la place de citer ici.

Merci enfin à Beethoven, Bauhaus, Mano Negra et tous les compositeurs et interprètes qui m'ont tenu compagnie et donné l'inspiration durant ces longues journées passées à rédiger.

Table des matières

Deuxième Partie
Comparaison de deux produits : Views et Spirit

Introduction

L'informatique documentaire est la partie de l'informatique qui s'occupe des documents, et les documents sont les supports privilégiés de l'information textuelle susceptible de faire l'objet de recherches. Au sein de cette introduction, nous allons nous efforcer de donner un aperçu du domaine à travers une typologie des systèmes informatiques de gestion documentaire.

Les **documents** sont définis ici comme des informations élémentaires (c'est-à-dire non structurantes) et spécifiques, généralement représentées sous forme écrite, graphique ou mixte[1]. Les documents sont souvent caractérisés par les propriétés suivantes : leur présentation est structurée, ils sont destinés à être archivés, et leur distribution se fait à l'aide de moyens de communication asynchrones (poste interne ou externe, par exemple). Plusieurs types de documents peuvent être distingués : on peut citer, entre autres, les lettres, les brochures, les rapports de réunions, les articles de revues et les livres.

Comme le montre la figure 0.1., la classe des **informations élémentaires** comprend, outre les documents, d'une part les messages, plus courts et sans structure particulière, et d'autre part les formulaires, qui sont tout à fait standardisés. La classe des **informations structurantes**, quant à elle, comprend les dossiers, les fichiers et les piles, qui, tous, **groupent** des occurrences d'informations élémentaires. Les dossiers regroupent messages, documents et formulaires par lien "logique" (d'ordre sémantique); les fichiers regroupent des formulaires univoquement identifiables d'un même type (parfois des messages ou des documents); les piles, elles, sont des "fourre-tout" : on y trouve tous les types d'informations, sans lien logique ni ordre intentionnel, si ce n'est que ces informations sont en attente et apparaissent généralement par ordre chronologique d'arrivée. Pour plus de précisions sur la modélisation du bureau, veuillez consulter le premier chapitre de [LESU90].

---

1 Cfr [LESU90], chapitre 1.

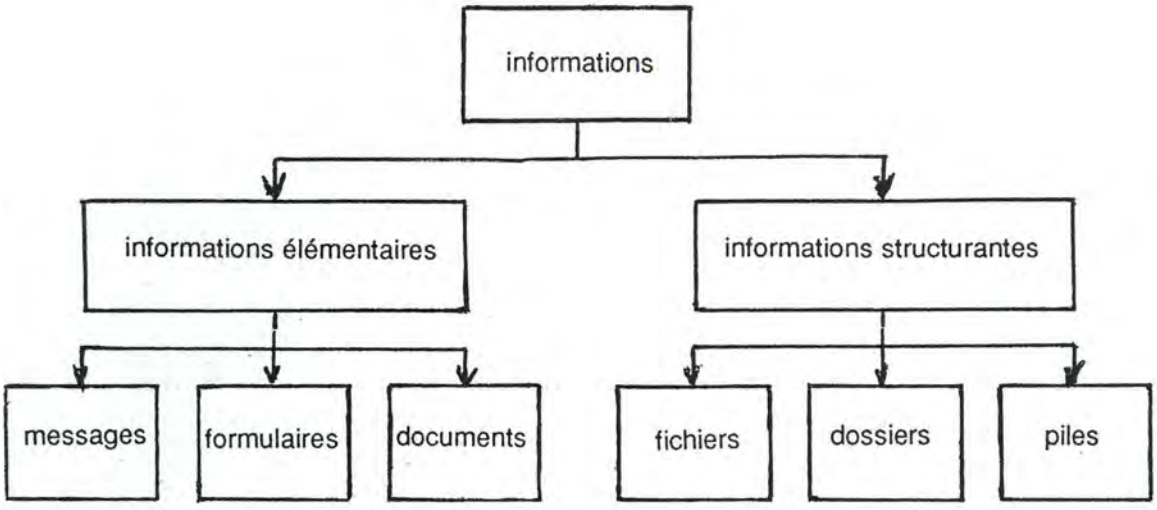

Figure 0.1. Les types d'information selon [LESU90]

Afin de bien cerner le cadre de ce mémoire, nous avons pensé qu'il est nécessaire de dresser un "tableau" des systèmes informatiques que l'on rencontre dans le domaine documentaire. Cette catégorisation des systèmes nous amènera à percevoir clairement les types d'enjeux et de problèmes rencontrés dans chacune des classes d'applications informatiques.

L'identification de la tâche informatisée nous fournit le critère de découpe principal. Wainwright et Francis ([WAIN86]) et Lesuisse ([LESU90]) distinguent cinq grands types de tâches :

- les tâches de transformation de l'information : écriture, dictée, frappe, impression, ...
- les tâches d'organisation de l'information : extraction, synthèse, analyse, ...
- les tâches de dispersion ou de communication de l'information : réunions, présentations, distribution, ...
- les tâches de rangement de l'information : enregistrement, indexation, ...
- les tâches de recherche de l'information : lectures, observation, ...

Dès à présent, nous nous limiterons aux **tâches de recherche** et, dans la mesure où elles y sont liées, aux **tâches de rangement** de l'information.

Deux activités sont à distinguer lorsqu'on étudie la recherche d'informations informatisées. Il y a d'une part la recherche du ou des documents contenant (ou pouvant contenir) l'information désirée, et, d'autre part, la recherche, au sein d'un document, de l'information elle-même.

1. Recherche de documents contenant l'information pertinente

L'objet de la recherche est, dans ce cas-ci, l'ensemble des documents pertinents, c'est-à-dire les documents dont l'indexation correspond à celle de la demande; cette recherche consiste en un tri rapide destiné à séparer les documents probablement pertinents des documents vraisemblablement non pertinents. Le résultat ne sera donc pas, en toute généralité, l'information elle-même, mais une référence à un document qui contient l'information, ou, mieux, ce document lui-même (ce dernier cas ne se présentant que lorsqu'on stocke dans le système l'entièreté de chaque document).

Le problème auquel nous sommes confrontés est un problème de représentation des documents. En effet, nous avons, d'un côté, les utilisateurs et l'ensemble de leurs requêtes, et, de l'autre, les documents et l'ensemble de leurs représentations. C'est l'expression de la requête, c'est-à-dire le langage dans lequel elle est formulée, qui établit la relation entre ces deux ensembles[1].

Pour affiner un peu plus notre typologie, nous pouvons distinguer dans le cadre de l'"information retrieval" (la dénomination anglaise du domaine) plusieurs types de systèmes. Selon le type de représentation des documents, nous avons d'une part les systèmes utilisant l'entièreté du texte du document (dits "full text"), et d'autre part ceux qui reposent sur des succédanés du texte.

1.1. Recherche de documents sur le texte complet

Dès que les moyens techniques l'ont permis (le volume des données a longtemps été une contrainte prohibitive), des systèmes sont apparus qui utilisent l'entièreté du texte du document pour la représentation. L'avantage de ces systèmes est la possibilité d'automatiser totalement le processus d'indexation des documents. Les problèmes sont d'ordres divers : volume des

---

1 On peut souligner que l'indétermination de ces deux ensembles est à l'origine de l'extrême complexité du domaine.

données, traitement du langage naturel, définition d'un "bon" critère d'appariement entre expressions de requêtes et documents, ...

1.2. Recherche de documents sur des succédanés du texte

A cause de la quantité et la taille des informations à stocker, il n'a pas tout de suite été possible de disposer de l'entièreté du texte pour chercher les documents. Il a fallu choisir les représentations des documents les plus économiques et les plus efficaces possibles.

La composante "économique" se mesure en termes du coût de stockage de l'information et du coût de traitement lors de la création des mécanismes d'accès. L'efficacité, quant à elle, se mesure lors de la recherche, à l'aide d'indicateurs de la pertinence des réponses[1] : le taux de précision ("precision" en anglais) est le rapport entre le nombre de documents pertinents donnés dans la réponse du système et le nombre total de documents dans cette réponse, et le taux de rappel ("recall" en anglais) est le rapport entre le nombre de documents pertinents donnés dans la réponse du système et le nombre total de documents pertinents dans la collection des documents. Nous reviendrons à ces mesures et à leurs dérivées par la suite.

Plusieurs expériences ont été menées[2], d'abord avec des "unitermes" (projet Cranfield II) : un (et un seul) descripteur était choisi par document, de manière à pouvoir mener des recherches sur mots-clés. Les résultats, quoique supérieurs aux recherches conventionnelles sur titre, auteur, etc., ne furent pas vraiment extraordinaires. Depuis lors, beaucoup d'approches nouvelles ont été conçues et réalisées. La plupart se basent sur des mots-clés assignés soit automatiquement soit manuellement, d'autres sur des résumés construits, à nouveau, tantôt par la machine tantôt par l'homme, et les plus récentes sur des représentations sémantiques des textes (frames, par exemple).

2. Recherche d'information au sein du texte d'un document

Après avoir identifié un document au sein duquel l'information cherchée a toutes les chances de se trouver, l'utilisateur est confronté à un second problème. Il s'agit maintenant de repérer au sein du texte parfois

---

1 L'efficacité prend aussi en ligne de compte le temps de réponse, le coût de l'information pertinente (surtout pour les systèmes on-line), etc.

2 Pour un historique détaillé et commenté, consultez [ELLI90] (la première partie). Une seule réserve vis-à-vis de cet excellent petit livre : seules les recherches menées dans le monde anglophone y sont mentionnées.

très long (songeons aux livres) le ou les passages intéressants, et ce, le plus rapidement possible, tout en étant certain de ne rien laisser de côté.

C'est pourquoi tout le texte du document devra être présent dans le système. Par ailleurs, nous verrons que le document comporte d'autres éléments que le texte (la structure, les figures, ...) dont les programmes informatiques doivent tenir compte. Ceci pose des problèmes de taille et de conventions.

Poursuivons l'élaboration de notre typologie. A ce niveau, nous ne pouvons plus utiliser le même critère de subdivision que pour la première classe. Ici, en effet, nous travaillons toujours sur le texte complet. Par contre, on peut opérer une distinction entre les systèmes selon l'expression des requêtes. La plupart des logiciels fonctionnent avec des requêtes booléennes, d'autres (beaucoup plus rares) proposent le langage naturel, tandis que les derniers implémentent les liens dits "hypertextes". A la différence des catégories précédentes, celles-ci ne sont pas exclusives. Un même logiciel pourra offrir plus d'un mode d'interrogation (nous verrons l'exemple de Folio Views).

2.1. Requête formulée à l'aide d'une expression booléenne

Le terme "expression booléenne" est faussement restrictif : on doit y inclure tout opérateur déterministe. Outre les opérateurs de présence ("et", "ou") et d'absence ("pas") ainsi que leurs dérivés ("ou exclusif", etc.), l'expression booléenne pourra contenir, par exemple, des opérateurs de proximité ("dans le même paragraphe", "à x mots de distance", etc.). Il est possible de diversifier ces opérateurs à loisir. Il ont tous pour dénominateur commun qu'ils s'inscrivent dans la logique déterministe du tiers-exclus : un passage conviendra si et seulement si le prédicat (à valeur vraie ou[1] fausse) défini par la requête est "vrai" pour ce passage.

Du fait de leur nature, ces opérateurs conviennent bien pour construire des prédicats du type expression régulière. Nous verrons sur un exemple (Folio Views) que c'est effectivement dans ce cadre qu'ils sont utilisés.

Nous pouvons déjà percevoir leur pauvreté en constatant que le texte ne doit être "compris" par la machine qu'à son plus bas niveau, c'est-à-dire

---

1 Il s'agit ici d'un "ou exclusif".

comme une suite de caractères, certains d'entre eux jouant le rôle de séparateurs (espace, ponctuation, etc.).

2.2. Requête formulée en langage naturel

Pour l'utilisateur, la formulation des requêtes en langage naturel signifie l'affranchissement par rapport à toute convention, à tout langage d'interrogation formel. La notion d'opérateur disparaît, pour laisser la place aux phrases ou aux concepts.

Quiconque a étudié un peu l'informatique sait que pour parvenir à la compréhension (même très partielle) d'une langue, les techniques logicielles sont très complexes. On peut les distinguer selon qu'elles sont basées sur l'analyse syntaxique ou sur l'analyse sémantique. La frontière entre ces deux approches est floue : certains produits sont à cheval sur les deux. Nous verrons dans la deuxième partie comment Spirit, conçu à l'origine exclusivement pour l'analyse linguistique s'oriente maintenant vers l'analyse sémantique du texte.

L'analyse syntaxique s'attache à reconnaître les règles de construction de chaque phrase, et d'en déduire des rôles que jouent les mots. Selon qu'un mot est pris dans tel ou tel cas comme substantif exclut la valeur sémantique qu'il aurait eue comme conjonction, par exemple (penser aux significations de "or").

L'analyse sémantique essaie de dégager d'un texte les idées qui y sont enfermées. Elle fait généralement usage de représentations internes favorables à ce genre de démarche (frames, réseaux sémantiques, scripts, etc.), directement inspirées de l'intelligence artificielle.

2.3. Liens hypertextes et recherche d'information textuelle

Le troisième mode de recherche d'informations fait appel à la notion d'hypertexte. Nous verrons évidemment en détail comment un texte peut être transcrit en réseau hypertexte. Ce qu'il convient de savoir à ce stade est qu'un réseau hypertexte est constitué de nœuds informationnels. Ils contiendront les morceaux du document informatisé. Ces nœuds sont connectés les uns aux autres par des liens que l'utilisateur peut employer pour naviguer d'un nœud vers l'autre.

Les liens sont de toute nature : tantôt ils enregistrent la structure du document, tantôt ils uniront deux paragraphes sémantiquement proches, tantôt ils fourniront à l'utilisateur le moyen de lire le document

autrement (s'intéresser uniquement aux citations du Grand Robert, par exemple).

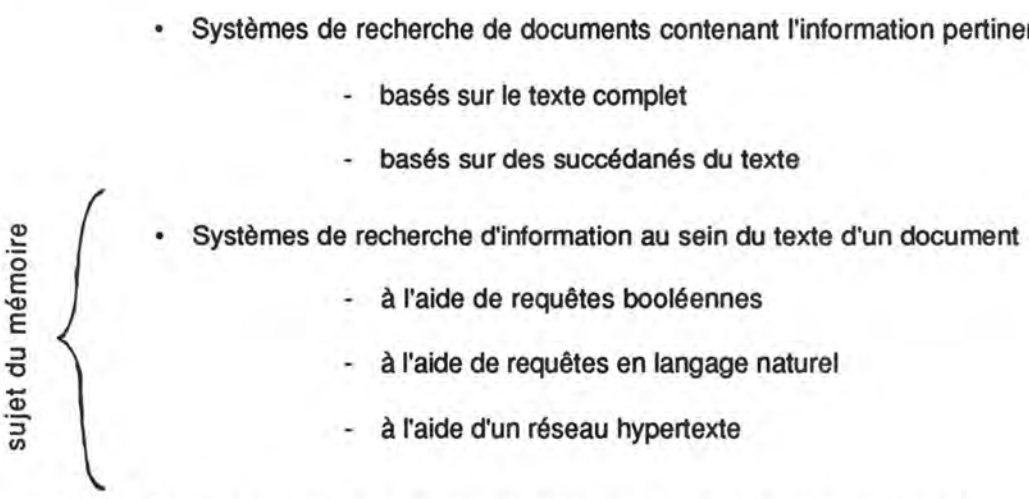

Figure 0.2. Les domaines de la recherche documentaire et le sujet du mémoire

Nous avons à présent terminé notre typologie, illustrée par la figure 0.2. Dans le cadre de ce mémoire, nous avons estimé qu'entreprendre une étude globale serait prétentieux. Nous avons donc restreint l'étendue du sujet aux seuls systèmes de stockage et de recherche d'information textuelle informatisée, c'est-à-dire à la deuxième grande catégorie de logiciels dont nous venons de parler[1]. Nous allons les présenter sous deux angles différents et complémentaires.

---

1 En ce qui concerne la première catégorie de systèmes, dits de recherche documentaire ou d'"information retrieval", nous ne pouvons que vous recommander la lecture des publications suivantes. Ces références sont évidemment non exhaustives, mais nous pensons qu'elles forment une bonne base de renseignements. [VANR79] et [SALT83] donnent un aperçu très clair et très précis du domaine. Ce sont les deux livres à lire en priorité; aucun des deux ne fait appel à des connaissances préalables, si ce n'est dans les parties parlant de statistiques où une maîtrise des éléments de base est indispensable. Plus général et beaucoup plus facile d'accès, [PAO89] est certainement plus indiqué pour un lecteur non informaticien, mais l'auteur va moins en profondeur. Nous avons déjà renseigné [ELLI90]; [TENO90] s'adresse au lecteur désirant accroître ses connaissances dans le domaine du full text retrieval. N'oublions pas les ouvrages "fondateurs" : [ZIPF49] et [LUHN58] restent des jalons, ainsi que, dans une moindre mesure, [LANC68] et [SPAR71]. Beaucoup plus récent et empruntant une voie nouvelle, [BLAI90] est sans aucun doute le livre le plus intéressant de cette petite liste. Néanmoins, il requiert de bonnes connaissances préalables en informatique documentaire. Nous en terminerons avec [SAFF89], qui offre une bibliographie commentée très complète; son seul défaut (mais ce livre est loin d'être le seul dans son cas) est de ne s'intéresser qu'à ce qui est publié dans le monde anglophone.
Du côté des périodiques, "ACM Transactions on Office Information Systems" et "Information Processing and Management" (revue anciennement dénommée "Information Storage and Retrieval") sont assurément les endroits où convergent les résultats de toutes les recherches. "The Communications of the ACM", "The Computer Journal" et "IEEE Computer" (difficilement trouvable !) reprennent généralement des articles publiés précédemment, éventuellement dans des compte-rendus de conférences, mais ne sont pas entièrement consacrés au domaine de la recherche documentaire. La revue "Technique et Science Informatiques", quoique non réservée au domaine, publie de bons articles et a l'avantage d'être écrite en français.
Malgré tout, la majorité des nouveautés apparaît en premier lieu dans les recueils d'articles présentés aux conférences. Citons, en vrac et sans ordre de préférence, "RIAO (Recherche d'Information Assistée par Ordinateur)", "Online information meeting", "Research and Development in Information Retrieval", "SIGIR (Special Interest Group on Information Retrieval)" et "SIGOIS" (Special Interest Group on Office Information Systems), "Informetrics", et "International Forum on Information and Documentation".

En premier lieu, nous expliquerons quels sont les concepts et les méthodes mis en œuvre pour réaliser des outils informatiques de stockage et de recherche d'information textuelle.

- Le premier chapitre donnera une idée précise de ce que l'on entend par unité d'information ou document.
- Le deuxième chapitre mettra en exergue toutes les informations d'un texte non incluses dans la suite des mots et montrera les techniques utilisées pour conserver ces informations.
- Le troisième chapitre détaillera les trois niveaux d'analyse du texte (lexical, syntaxique et sémantique), et les concepts qu'utilise chaque méthode.
- Le quatrième chapitre s'intéressera aux résultats des requêtes, à leur pertinence et aux moyens de l'évaluer.
- Le cinquième chapitre fera le point sur les méthodes de compression disponibles pour les données textuelles.
- Le sixième et dernier chapitre introduira quelques notions d'interfaces homme / machine et dégagera les options recommandables dans notre contexte.

En deuxième lieu, nous verrons, à l'aide de deux logiciels, Folio Views et Spirit, et d'une petite expérience menée sur un texte homéopathique, comment on passe de la théorie à la pratique.

- Le septième chapitre sera une introduction consacrée au livre et à la démarche adoptée lors de l'expérience.
- Les huitième et neuvième chapitres présenteront chacun un des deux produits et leurs spécificités, en illustrant les notions théoriques de la première partie.
- Le dixième chapitre, enfin, nous verra nous aventurer à faire une comparaison des deux logiciels, en prenant tour à tour les points de vue du concepteur (ou administrateur) et de l'utilisateur.

Première Partie

Concepts et méthodes mis en œuvre pour réaliser des outils informatiques de stockage et de recherche d'information textuelle

Chapitre 1.
L'unité d'information

Nous pensons qu'il est essentiel de comprendre la notion de document lorsqu'on aborde l'informatique documentaire. En effet, on se rend vite compte que chaque auteur donne sa définition (quand il la donne...), et que personne ne s'est mis d'accord. Il est clair que nous n'allons pas apporter de solution miracle au problème; tout au plus nous contenterons-nous de montrer les endroits où les vues sont divergentes.

Commençons par le début et voyons l'étymologie du mot "document", telle qu'elle figure dans [BLOC89].

> *"XII^{e}. Emprunté du latin "documentum" (de "docere") «enseignement, ce qui sert à instruire», seul sens du mot français jusqu'au XVII^{e} siècle; le sens moderne, qui paraît être issu de l'emploi de "document" comme terme de palais dans "titres et documents" depuis la fin du XVII^{e} siècle, date du début du XIX^{e}. - Dérivés : documentaire (1876), documentation (1870), documenter (1769)."*

La connotation informationnelle liée au mot est évidente. Passons à un dictionnaire ordinaire, le Grand Robert ([ROBE85], vol. 3, p. 610) :

> *"Ecrit servant de preuve ou de renseignement; par ext. «toute base de connaissance, fixée matériellement, susceptible d'être utilisée pour consultation, étude ou preuve» (Union Française des Organismes de Documentation)."*[1]

Au départ, le document contient de l'information écrite, textuelle. Plus généralement, on admet que cette information soit d'autre nature que le texte (on pense aux documents sonores), et trouve des supports matériels divers. Il semble bien que le document électronique corresponde à cette définition et soit ainsi le dernier-né d'une grande famille.

---

1 Le Grand Robert donne plusieurs autres définitions, moins intéressantes dans le cadre de notre propos.

Nous ne retiendrons que les documents textuels. Cependant, il ne faut pas sous-estimer l'importance sans cesse croissante des données non textuelles au sein des documents; les outils bureautiques devront s'adapter et proposer des solutions au problème du multi-média.

La consultation des documents et la recherche de l'information qui s'y trouve suppose une réponse. Cette réponse sera un ensemble de passages du document, pertinents pour la question posée. Nous approfondirons plus tard les notions de question et de pertinence. Attardons-nous maintenant sur ce qu'on entend par "passage".

Chacun des passages composant la réponse sera pour l'utilisateur une unité de réponse. Cependant, la taille et les débuts et fins de ces unités de réponse varient selon l'utilisateur et selon la question posée. Par exemple, la plupart des lecteurs considèrent que l'unité de réponse d'un dictionnaire est l'article. Ce n'est pas le cas de celui qui cherche toutes les citations de Montaigne.

Quand le document est sur papier, le lecteur définit lui-même son unité de réponse. Lorsqu'on passe au document électronique, c'est à la conception que la découpe se fera. Le résultat de cette découpe sera un ensemble de passages, sensé satisfaire tout le monde. Ces passages, appelés unités d'information, devront correspondre au mieux à ce que l'utilisateur attend comme unité de réponse. Certains logiciels permettent la superposition de plusieurs découpes; dans ce cas, l'utilisateur a le choix (limité) parmi les types d'unités de réponse proposés.

Par abus de langage parmi les informaticiens, l'unité d'information, c'est-à-dire ce que le concepteur considère comme l'élément de base du système documentaire, est souvent dénommé "document". Nous essaierons autant que possible d'éviter les malentendus en employant "unité d'information" ou "passage"; il peut arriver que nous fassions un usage abusif du terme "document", auquel cas nous le signalerons.

Chapitre 2.
Informations complémentaires : structure et liens

Nous venons de le voir, le document est destiné à être consulté et étudié, ce qui implique des moyens d'accès à l'information. En fait, les moyens d'accès sont présents dans chaque document, soit au sein du texte (l'introduction détaille le plan et la démarche de l'auteur), soit par la structure du texte (chaque paragraphe correspond à une idée, le titre reflète le contenu du texte qui suit), soit encore par la disposition, par la forme du texte (un changement de police de caractères indique les citations).

Quand un document est mis sous forme électronique, le risque est d'éliminer ces repères, au profit, il est vrai, d'outils plus performants mais moins familiers. Les indications figurant dans le texte ne posant aucun problème, nous verrons d'une manière approfondie les différents éléments d'un document à l'aide de la norme ODA de l'ISO. Nous décrirons ensuite l'approche hypertexte, dont la particularité est justement de mettre l'accent sur la liberté d'associer et de structurer l'ensemble des unités d'information.

2.1. ODA et la structure du texte[1]

L'élaboration d'une norme internationale est souvent l'occasion de réfléchir en profondeur aux écueils parsemant un domaine, et la solution récolte les fruits de cette réflexion. En 1986, l'ISO publiait[2] une norme de formalisation, de description des documents, ODA (Office Document Architecture). Nous pensons qu'il est bon de se pencher sur les concepts centraux d'ODA pour cerner au mieux le concept de document. D'autres normes

---

1 Cet exposé est inspiré de [ANDR90].

2 Cf. document ISO : Information Processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format (ODIF), ISO / DIS 8613, 1986.

existent, comme SGML[1]; nous avons choisi ODA pour sa complétude : il prend en compte tant la structure physique que la structure logique des documents.

Ouvrons un livre, un roman par exemple, et décrivons ce que nous y trouvons. A part celles du début et de la fin[2], les pages se ressemblent. Elles partagent le même format, les mêmes marges (appelées blancs de couture, de grand fond, de tête et de pied) et donc le même rectangle d'empagement. Les pages sont formées de pavés, chaque pavé étant lui-même formé de lignes de texte. Ces lignes peuvent être de plusieurs types : on aura, entre autres, les lignes en majuscules, précédées et suivies de grands espaces blancs, centrées (il s'agit des lignes de titre des chapitres), et les lignes "normales", justifiées, précédées ou suivies de blancs s'il s'agit du début ou de la fin d'un paragraphe.

En formalisant, on peut arriver aux règles suivantes, qui nous donnent la définition de la mise en page du document, appelée structure physique.

```
roman          = page+
page           = pave+  numero
pave           = ligne+ | grostitre
ligne          = signe+
grostitre      = majuscule+
numero         = chiffre+
signe          = majuscule | minuscule | espace | ponctuation
chiffre        = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
majuscule      = A | B | C | ... | Y | Z
minuscule      = a | b | c | ... | y | z
ponctuation    = . | , | ; | : | ? | ...
espace         =
```

où "=", "|", "+" et "A B" signifient respectivement "est", "ou", "une ou plusieurs fois" et "A suivi de B".

L'exemple que nous avons pris est simple à l'extrême; la mise en page d'un quotidien s'avérera beaucoup plus complexe. En outre, rien encore ne nous permet d'exprimer qu'un roman ne peut commencer que par un titre, et pas par un paragraphe de texte. Il faut donc utiliser d'autres notions : la séquentialité (deux éléments se suivent), la hiérarchie (un élément de structure n'existe qu'à l'intérieur d'un autre et la collatéralité (deux éléments sont indépendants).

Chaque composant d'ODA peut avoir les caractéristiques suivantes :

---

1 Cf. document ISO : Information Processing - Text and Office Systems - Structured Generalized Markup Language (SGML), ISO 8879, 1986.

2 Elles correspondent à la notion logique (par opposition à physique) de péritexte. Pour un ouvrage complet sur le péritexte et, plus généralement, sur le paratexte, consultez [GENE87].

- obligatoire : il doit apparaître une fois et une seule (noté "REQ")
- facultatif : il peut apparaître une fois ou pas du tout (noté "OPT")
- répétable : il peut apparaître une ou plusieurs fois (noté "REP").

Les attributs "OPT" et "REP" peuvent être combinés pour indiquer qu'un élément est à la fois facultatif et répétable (noté "OPT REP"). D'autres attributs indiquent la façon d'assembler les composants :

- en séquence, dans l'ordre indiqué (noté "SEQ")
- par agrégation, sans ordre déterminé (noté "AGG")
- au choix, un parmi les composants possibles (noté "CHO").

Nous venons, imperceptiblement peut-être, de glisser vers la notion de structure logique d'un texte. En effet, dire qu'un ensemble de lignes (non nécessairement sur la même page) forme une entité, un paragraphe, ou qu'une sous-section ne peut figurer que dans une section, relève de la logique du document. La structure logique est concernée par les relations entre les éléments d'un texte et la structure physique par leur représentation.

Les règles fournissant les deux structures seront différentes; pour le roman, nous pourrions, en adoptant les mêmes conventions, avoir la "grammaire" logique suivante :

```
roman          = chapitre+
chapitre       = paragraphe+
paragraphe     = texte
```

Les structures logique et physique ont trait au même document; elles se rejoignent donc au niveau de son contenu. La figure 2.1. montre comment les structures logique et physique de notre exemple se chevauchent[1]. Seules les deux premières pages sont reprises, mais la même structure se répétera[2]. Notez qu'en toute généralité, une page pourra contenir plusieurs cadres.

---

1 Les dénominations ont été simplifiées dans un souci de clarté. On pourrait enrichir le schéma pour montrer l'entièreté des deux structures.

2 On distingue d'ailleurs structure générique et structure spécifique. Ce qui est dessiné est la structure spécifique; la structure générique décrit les règles de construction des structures spécifiques (tant logiques que physiques), à l'aide des attributs que nous avons montré.

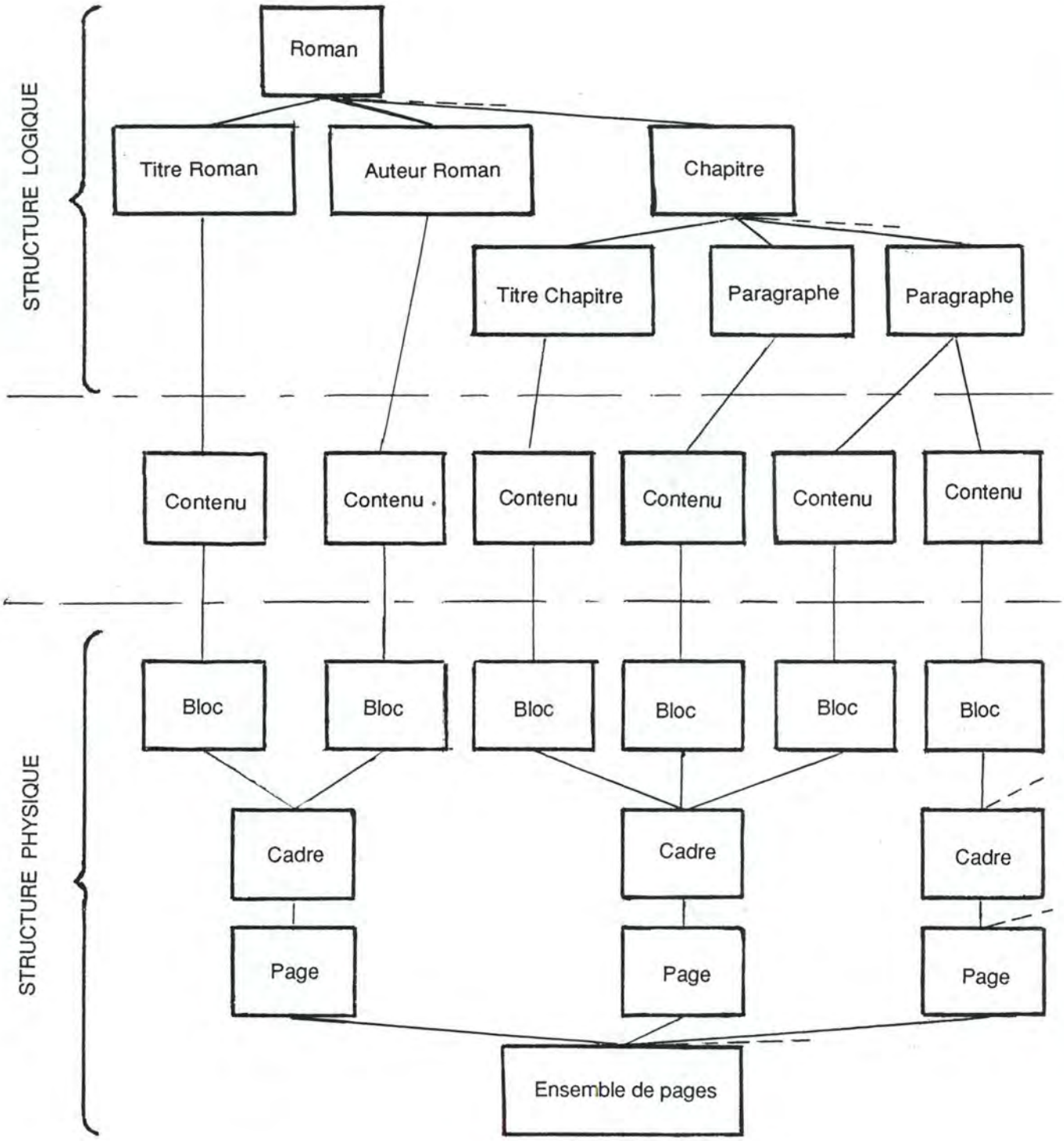

Figure 2.1. Correspondance entre structures logique et physique (spécifiques)

A une structure logique peuvent correspondre plusieurs structures physiques qui varieront selon les choix de mise en page. De la même manière, on peut associer à une structure physique plusieurs structures logiques valides, selon la sémantique donnée aux éléments de mise en page.

Pour en avoir terminé, il reste à dire que ce sont les feuilles des deux structures auxquelles est attaché le contenu qui pourra être de toute nature. Nous pourrions rencontrer, entre autres, des caractères, des graphiques photographiques ou des graphiques géométriques.

2.2. Les hypertextes

Nous venons de voir qu'un document est un objet dont les éléments sont liés par une structure arborescente. Il y a pourtant des cas où le graphe des liens n'est plus un arbre. On pense tout de suite aux figures, qui sont référencées plusieurs fois dans le texte, mais on peut aussi envisager le cas des renvois vers un autre passage (qui se trouve tout autre part dans la structure textuelle, à la fois horizontalement et verticalement).

La solution à ce problème de structure non hiérarchique, c'est l'approche hypertexte qui nous la donne. Notre objectif n'est certainement pas de faire ici un exposé complet sur les hypertextes[1], mais plutôt de montrer les concepts mis en œuvre pour modéliser le document.

2.2.1. Nœuds, liens et réseau hypertexte

Nous venons de voir qu'un document peut être perçu non seulement comme une suite linéaire de lignes ou de caractères formant les mots et les phrases, mais aussi comme un ensemble d'unités d'information structurées par des liens hiérarchiques, ou même par des liens non hiérarchiques. Nous reprenons ces notions dans le cadre des hypertextes, et nous les baptisons nœuds et liens.

"Les nœuds sont des contenants d'information considérés par l'utilisateur qui les crée comme des entités sémantiques distinctes les unes des autres. Un lien représente une relation entre nœuds ou entre des informations contenues dans les nœuds" [DANI90]. L'ensemble des liens et des nœuds forme le réseau hypertexte; on peut schématiser ce réseau à l'aide d'un graphe où les sommets sont les nœuds et les arcs les liens. Vous trouverez un exemple de réseau à la figure 2.2.

Les nœuds s'apparentent aux unités d'information d'un document, et les liens à la structure textuelle. Cependant, les hypertextes vont plus loin. L'information contenue dans les nœuds n'est pas nécessairement textuelle, et les liens peuvent rendre compte d'une structure non

---

1 A cette fin, consultez les ouvrages renseignés par [SAVO90], spécialement [CONK87] et [DANI90].

hiérarchique. En outre, rien n'empêche de relier deux nœuds appartenant à des documents différents; la notion de document perd alors un peu de son sens, puisque les frontières disparaissent.

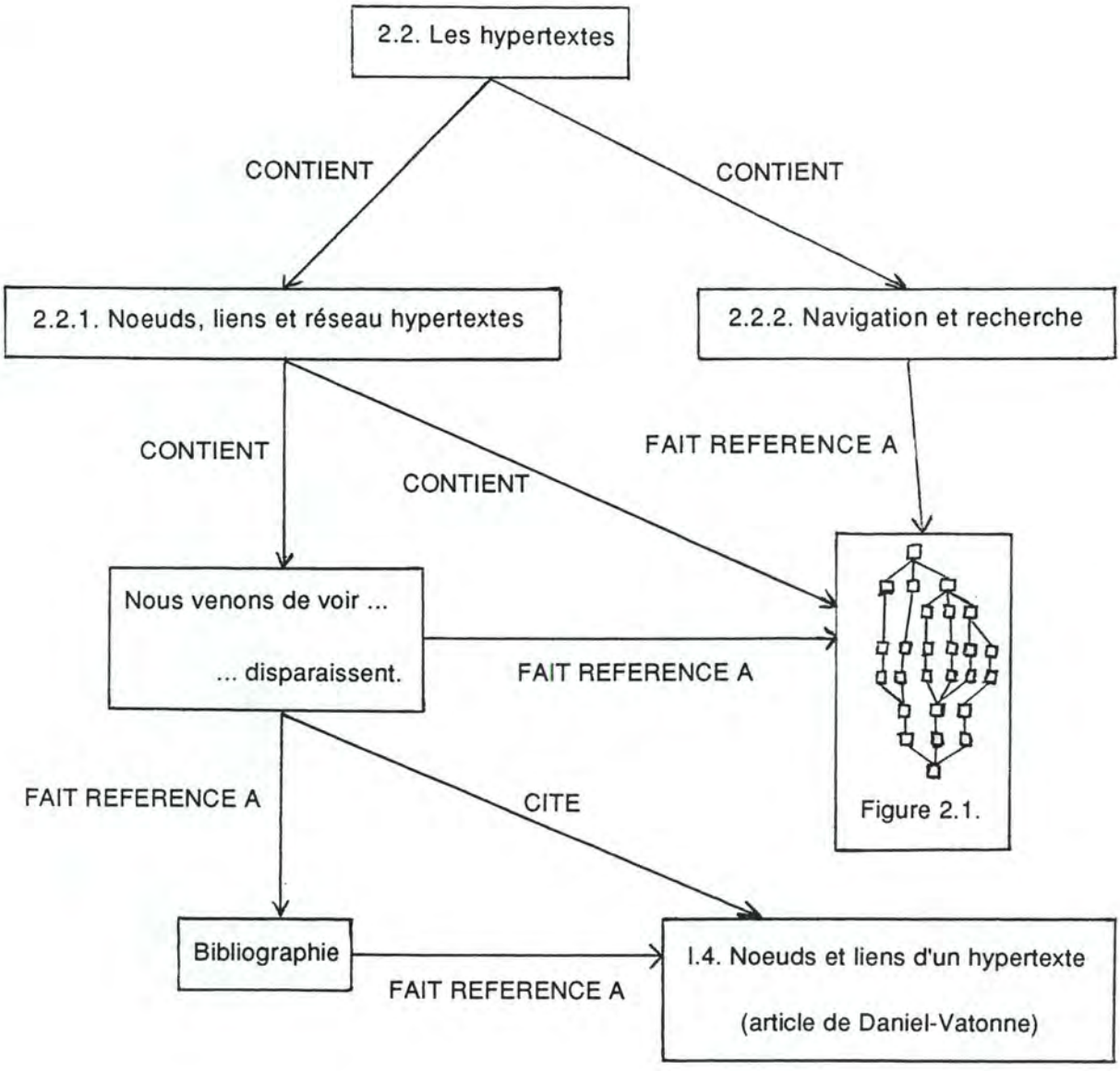

Figure 2.2. Un exemple de réseau hypertexte

2.2.2. Navigation et recherche

Les nœuds sont destinés à être affichés à l'écran, et les liens permettent à l'utilisateur de parcourir les différentes parties du document. Ces liens doivent être explicites, faciles à suivre, et doivent transporter du nœud référent (celui d'où sort le lien) au nœud référé (celui où l'extrémité aboutit) rapidement. Un lien est présenté à l'utilisateur par

des icones de lien (dits "boutons"). La figure 2.2. illustre des boutons "contient", "fait référence à" et "cite".

Les deux ancres d'un lien sont les endroits d'où part et où arrive le lien. Le point de départ peut être plus petit qu'un nœud : il s'agira le plus souvent d'un élément du nœud, comme un mot ou une icône, c'est pourquoi la représentation qu'on a donnée n'est pas idéale.

Le principe de la navigation est de parcourir le réseau en utilisant les liens pour atteindre et consulter les nœuds. Afin de faire apparaître le nœud à l'autre extrémité du lien, il s'agira, le plus souvent, de cliquer avec la souris sur l'ancre de départ[1]. Une autre façon de naviguer est de pointer sur la carte du réseau l'endroit où on veut se déplacer. La carte du réseau est habituellement disponible, et constitue un moyen d'orientation au sein du réseau qui est généralement très grand.

Une caractéristique essentielle des hypertextes est la navigation : les "documents" (utilisons plutôt le terme "réseau" dans ce contexte, il est plus approprié) sont consultés dans un ordre qui n'est pas linéaire. L'utilisateur, à tout moment, fait le choix de continuer à lire séquentiellement, de passer à un nœud sémantiquement proche, de consulter la carte, etc. Le gros problème des hypertextes est lié à cette multitude de choix : on oublie le chemin suivi et on perd la perception de l'endroit où on se trouve. Par contre, la grande force de la navigation, c'est la liberté offerte à l'utilisateur d'avoir accès à toute l'information, d'y faire son chemin, et d'y trouver la réponse sans devoir formuler une question. Cet aspect est particulièrement important lorsque sa connaissance du sujet et du réseau est limitée.

L'autre méthode d'exploration du réseau, la recherche, peut se faire soit sur le contenu des nœuds (le plus souvent avec des requêtes booléennes) soit sur leur structure (par exemple, chercher tous les nœuds isolés). Il ne s'agit plus de suivre les liens mais d'afficher directement un nœud ou une partie du réseau.

---

1 La souris et la manipulation directe sont les piliers de la plus grande majorité des systèmes hypertexte.

2.2.3. Types de nœuds, de liens et de structures

Comme nous l'avons déjà laissé entendre, il y a deux grands types de nœuds[1] : les nœuds information et les nœuds composés. Les nœuds information sont définis comme des structures ayant un contenu informationnel et des attributs. Ces attributs, variant selon les systèmes, caractérisent le plus souvent la présentation du nœud (nature de l'information et modalités d'affichage), son identification (nom, auteur, date de création, etc.) et sa situation dans le réseau (énumération des liens entrants et sortants). Les nœuds composés servent à grouper les nœuds information afin de structurer le réseau. Ils fournissent des vues partielles du réseau et les tables des matières.

Les liens aussi forment deux grandes familles. Les liens hiérarchiques traduisent les structures logiques des documents; ils relient les nœuds composés aux nœuds qu'ils représentent et forment une arborescence plane et orientée (l'ordre est défini par l'ordre de lecture). Les liens de référence sont utilisés dans tous les autres contextes : ils servent à relier les nœuds dont le contenu sémantique est proche, les nœuds liés les uns aux autres par des renvois, des notes, etc. Ils fournissent un ordre de lecture non linéaire et leur représentation sous forme de graphe n'a aucune propriété particulière.

La structure primaire d'un réseau hypertexte est obtenue en ne tenant compte que des liens hiérarchiques. En revanche, la structure secondaire est celle qui se superpose à la structure primaire d'un réseau hypertexte (représentant un seul document) correspond à la structure logique qu'on a vue dans ODA. Dans l'approche hypertexte, la structure secondaire est mise en exergue, au point de devenir prépondérante.

---

1 Au départ, cette distinction n'était opérante que pour NoteCards; elle est en voie de se généraliser.

Chapitre 3.
Analyse du texte et briques de base des requêtes

Jusqu'à présent nous avons montré que le document est décomposé en unités d'information, qui seront les unités de réponse du système. Nous avons aussi vu que des liens de type structurel ou sémantique assurent la cohésion entre les unités informationnelles.

Préoccupons-nous maintenant de voir comment une unité informationnelle est stockée dans le système informatique, et comment on pourra formuler les requêtes, compte tenu du format interne des données. Le texte peut faire l'objet de trois types d'analyse, qui déterminent les "briques de base" textuelles et donc le type de requêtes qu'il sera possible de construire.

L'analyse lexicale détecte les formes[1], procède dans certains cas à une suppression des séquences terminales[2], et les seuls opérateurs envisageables sont booléens (éventuellement "étendus" à l'aide des probabilités) ou positionnels. L'analyse syntaxique dégage les mots du texte et leur valeur grammaticale. Les relations entre mots ne sont pas booléennes : on parlera plutôt de dépendance syntagmatique; les requêtes pourront être formulées en langage naturel. L'analyse sémantique a pour objectif l'extraction du sens; elle fait appel à des représentations internes empruntées à l'intelligence artificielle et permet d'arriver à des dialogues du style question / réponse.

3.1. Analyse lexicale

Un texte est perçu au niveau lexical comme une suite finie de caractères puisés parmi un alphabet fini. Un alphabet (parmi d'autres) comprendrait les lettres (a..z et A..Z), les chiffres (0..9), les signes de ponctuation (,;:?.()!-+=/'"), et l'espace (noté •). Un texte (parmi d'autres) construit à partir de cet alphabet est :

L'autre•petit•chat•est•mort aujourd'hui.

---

1 Une forme est une séquence de caractères délimitée par des espaces ou des signes de ponctuation.

2 Une séquence terminale est une suite de caractères située en fin de forme et jouant ou pouvant jouer le rôle de suffixe. Par exemple, "er" est la séquence terminale des verbes du premier groupe.

Le but de l'analyse lexicale est de dégager de cette suite de caractères les formes qui la composent. Une forme est une suite de caractères délimitée par des espaces ou autres séparateurs.

Il faut faire la différence entre les caractères qui composent les formes (les caractères concordables) et les séparateurs de mots (les caractères non-concordables). Il y a encore les semi-concordables : ce sont les caractères qui tantôt tomberont dans la première catégorie, tantôt dans l'autre. Une forme est alors définie comme une suite de caractères concordables ou semi-concordables bordée par des caractères non concordables ou semi-concordables.

Nous obtenons donc les formes "L", "autre", "petit", "chat", "est", "mort", "aujourd" et "hui", et constatons que la découpe n'est pas idéale : l'apostrophe, caractère semi-concordable, ne peut être analysé correctement sans recourir au dictionnaire[1].

D'autre part, si on trouve, dans une autre phrase, la forme "chats", ne s'agit-il pas du même concept, et n'y a-t-il pas un moyen simple de dégager ce concept ? La plupart des analyseurs lexicaux disposent de tables de séquences terminales et extraient (tant bien que mal[2]) les racines des formes. On peut aussi proposer, à l'interrogation cette fois, des caractères appelés jokers. Par exemple, "?", "*n" et "*" pourront remplacer un, i (où i appartient à l'intervalle [0, n]) ou un nombre illimité de caractères, notamment à la fin des formes.

Les opérateurs qui conviennent à la recherche de formes sont les opérateurs booléens. Par exemple, on cherchera les unités d'information contenant à la fois "chat*1" et "mort*1" en se servant du "et" booléen. On a constaté, et cela s'est vérifié au cours de notre expérience, que le "et" utilisé sans jokers terminaux est particulièrement traître : seul un petit nombre des passages pertinents vérifie le prédicat exprimé dans la requête. La figure 3.1. donne un tableau des opérateurs booléens les plus répandus.

A and B	Toutes les unités d'information retrouvées doivent contenir les deux opérands A et B. Cette opération est utilisée pour restreindre le domaine de la recherche par la présence simultanée des deux opérands.

---

1 Le résultat n'aurait pas été meilleur (loin s'en faut !) si l'apostrophe avait été pris comme caractère concordable.

2 On bute vite sur des exceptions : "suis" est-il issu de "suivre" ou de "être" ?

A or B	Toutes les unités d'information retrouvées doivent contenir l'opérand A ou l'opérand B ou les deux. Cette opération est utilisée pour élargir le domaine de la recherche en regroupant des termes similaires.
A xor B	Toutes les unités d'information retrouvées doivent contenir soit le premier, soit le deuxième opérand, mais pas les deux.
not B	Toutes les unités d'information retrouvées peuvent contenir n'importe quoi sauf B (opérateur unaire).
A not B	Toutes les unités d'information retrouvées doivent contenir A mais pas B.

Figure 3.1. Opérateurs booléens

Les opérateurs positionnels peuvent aussi être utilisés avec les formes; ils permettent de formuler des conditions sur la position relative des formes. On peut, par exemple, demander que "chat" (ou "chats") et "mort" (ou tout autre mot commençant par "mort") soient dans un voisinage de cinq mots maximum : "chat*1 near5 mort*". La liste des opérateurs de proximité les plus répandus[1] fait l'objet de la figure 3.2. Notons que les notions de phrase et de paragraphe qu'ils utilisent sont définis à un niveau purement lexical : un paragraphe sera délimité par deux caractères séparateurs de paragraphes, et il en va de même pour la phrase.

A adj B	Dans toutes les unités d'information retrouvées, A et B doivent être adjacents et dans l'ordre A, B.
A adjN B	Dans toutes les unités d'information retrouvées, A et B doivent figurer dans la même phrase dans l'ordre A, B avec un maximum de N mots les séparant.
A near B	Dans toutes les unités d'information retrouvées, A et B doivent être adjacents, dans n'importe quel ordre.
A nearN B	Dans toutes les unités d'information retrouvées, A et B doivent figurer dans la même phrase, dans n'importe quel ordre, avec un maximum de N mots les séparant.
A with B	Dans toutes les unités d'information retrouvées, A et B doivent appartenir à la même phrase.

---

1 Nous avons repris les opérateurs proposés par STAIRS (produit IBM).

A notwith B	Dans toutes les unités d'information retrouvées, A et B doivent figurer dans des phrases différentes.
A same B	Dans toutes les unités d'information retrouvées, A et B doivent appartenir au même paragraphe.
A notsame B	Dans toutes les unités d'information retrouvées, A et B doivent figurer dans des paragraphes différents.

Figure 3.2. Opérateurs de proximité

La sécheresse, le caractère déterministe des opérateurs que nous venons de décrire a motivé la recherche d'un appariement plus nuancé entre requêtes et unités d'information. Les systèmes probabilistes et statistiques sont alors apparus; ils reposent tous sur une fonction de similitude, calculée à partir de plusieurs paramètres, tels que le nombre de fois que les mots de la question figurent au sein du passage, au sein de l'entièreté du document, etc.

3.2. Analyse syntaxique

L'analyse syntaxique dégage la **structure** des phrases, c'est-à-dire les liens qu'ont les mots entre eux. Dans notre exemple, il faudra identifier les syntagmes nominal et verbal ainsi que l'adverbe, et voir que trois adjectifs se rapportent au même nom, quoiqu'il y ait deux épithètes et un attribut.

Prenons deux autres phrases pour montrer la complexité du problème.

La force brute (est fascinante)
La brute force (l'entrée)

Dans les deux cas, on a les mots "force" et "brute", mais dans chacun des deux cas ils ont des valeurs, des fonctions différentes (ils appartiennent à des classes grammaticales différentes).

Dans un premier temps, il s'agit de formaliser les règles qui président à la construction des phrases. Cela peut être réalisé à l'aide de la notation BNF (Backus-Naur Form). Voici un exemple de grammaire :

```
P ::    SN  SV
SN ::   A  N |
        A  D  N |
        N
SV ::   V  SN
```

où les abréviations ont les significations suivantes :

```
P :     Phrase
SN :    Syntagme Nominal
SV :    Syntagme Verbal
A :     Article
D :     Déterminant
N :     Nom
V :     Verbe
```

A l'aide de cette grammaire, on peut déjà construire des phrases telles que :

Le petit garçon mange la pomme

L'analyse syntaxique va identifier les mots de la manière suivante :

```
Le :                            A
Petit :                         D
Garçon :                        N
Le petit garçon :               SN
mange :                         V
la :                            A
pomme :                         N
la pomme :                      SN
mange la pomme :                SV
Le petit garçon mange la pomme :  P
```

L'exemple est évidemment simpliste, et il est clair que la définition exhaustive d'une langue parlée[1] est très complexe à réaliser.

La grammaire utilisée dans l'exemple fait partie des grammaires sans contexte (context-free grammars). Ces grammaires construisent un arbre syntaxique[2] en décomposant la phrase en éléments plus simples à l'aide des règles et de dictionnaires. Elles sont relativement faciles à construire et suffisent dans la majorité des cas, mais elles présentent néanmoins de sérieux inconvénients.

D'abord, toute phrase dont la structure n'a pas été prévue ne sera pas prise en compte, et, au pire des cas, ne sera pas indexée correctement. Ensuite, toute phrase nécessitant une notion de contexte pour être analysée échappera au système. Sans aller à l'ambiguïté parfaite de "la petite brise

---

1 Ou même d'un sous-ensemble de la langue.

la glace", des phrases comme "time flies like an arrow"[1] en anglais, peuvent mener à des aberrations si l'on ne tient pas compte du contexte. Les quatre interprétations que Salton et McGill donnent à cette phrase sont : (1) le temps passe aussi vite qu'une flèche, (2) on doit chronométrer les mouches comme une flèche chronomètre les mouches, (3) on doit chronométrer les mouches qui ressemblent à une flèche, et (4) il y a une espèce de mouches, les mouches du temps, qui aiment une flèche.

Mais, par dessus tout, la seule structure des phrases ne permet pas toujours de dégager les concepts qui s'y trouvent exprimés. Une figure de style permettra par exemple de séparer syntaxiquement des termes qui sémantiquement forment une entité. Il est souhaitable, pour augmenter l'efficacité de l'analyse syntaxique, de disposer d'un indicateur des corrélations que certains éléments du langage partagent avec d'autres.

Il y a donc eu d'autres formes d'analyse syntaxique qui ont vu le jour. D'abord on a vu apparaître les grammaires transformationnelles, qui ont amélioré le traitement standard des grammaires sans contexte (dont le résultat fut dès lors appelé structure surperficielle) avec des batteries de transformations pour obtenir la structure profonde du texte (deep structure). Puis sont arrivées les grammaires basées sur les automates finis (finite state machines) et sur les réseaux de Pétri enrichis. Les places correspondent aux états d'avancement de l'analyseur et les transitions aux symboles apparaissant dans le texte. Comme les symboles peuvent être ambigus, il est intéressant d'affecter des probabilités de réussite (a priori) à chacune des transitions, de telle manière à alléger le processus. Un mécanisme de backtracking est implémenté, afin de ne pas passer à côté d'une découpe valide après s'être engagé sur une fausse route.

Pour finir, j'aimerais souligner que Chomsky [CHOM57] a démontré que tout processus d'analyse syntaxique est voué à l'incomplétude (au moins sur une machine dont les ressources en mémoire sont finies). Il y a toujours moyen de construire des phrases imbriquées du style "Marc a rencontré celui qui a rencontré celui qui a rencontré ... Alain hier". L'entièreté de la

---

2 Chomsky a été un des pionniers en ce domaine de la recherche en linguistique. On peut se référer à [CHOM65] pour un exposé complet.

1 Exemple repris de [SALT83].

phrase devra être analysée avant de révéler l'ambiguïté du mot "hier", qui n'est pas explicitement attaché à telle ou telle subordonnée[1].

Lorsqu'on dispose d'un analyseur syntaxique, on va le faire travailler tant sur le texte des unités d'information que sur la question, formulée maintenant en langage naturel. Il procédera à la levée des ambiguïtés et à l'analyse des relations entre les mots pour détecter quels sont les passages les plus proches de la question. La notion d'opérateur perd évidemment son sens, étant donné le niveau plus élevé de "compréhension" du texte par la machine.

3.3. Analyse sémantique

L'analyse la plus poussée, l'analyse sémantique, va néanmoins aller plus loin, le but ultime étant d'extraire la signification du texte. Ce processus d'analyse sémantique pourra lui aussi être raffiné à des degrés divers. Le modèle le plus simple de la connaissance est sans doute le thésaurus organisé sémantiquement. Trois règles en régissent la construction : (1) les mots les plus fréquents du langage auront tendance à être situés à la racine du thésaurus, (2) les mots les moins fréquents auront tendance à être situés dans les feuilles, aux extrémités du thésaurus, et (3) les mots qui ont tendance à figurer ensemble devront être proches dans l'arbre.

Un autre modèle sur lequel on peut baser l'analyse sémantique est celui des catégories sémantiques. Chaque mot reçoit une description composée de quelques attributs, éventuellement multivalués.

Encore plus pour l'analyse sémantique que pour l'analyse syntaxique, les limites théoriques inhérentes au domaine contraignent à quelques réserves. En fait, le problème majeur est de définir la notion de sémantique, de sens. Une définition opérationnelle est de dire qu'une assertion a du sens si et seulement si un nombre fini d'observations permet de la considérer comme vraie ou fausse. Que devient la grand-mère à roulettes de M. Cherton, dans cette perspective ? Et les tautologies ? Et les phrases du type "tous les hommes sont mortels" ? Et quand bien même on admettrait que

---

1 Les ambiguïtés sémantiques échapperont évidemment à l'analyse syntaxique : la livre (Sterling) et la livre (de beurre) sont confondues. Seules les langues non ambiguës comme l'esperanto et les sous-langages peuvent faire l'objet d'une levée d'ambiguïtés complète.

tout cela n'a pas de sens, il y a toujours les paradoxes, et ce théorème de Gödel[1]...

---

1 Gödel a prouvé en 1931 l'incomplétude de tout système formel incluant la théorie des nombres.

Chapitre 4.
Résultat des requêtes

Nous avons vu que le document est découpé en unités informationnelles, qui constituent les éléments de réponse du système. Après avoir montré la manière de formuler les questions, nous allons examiner le résultat des requêtes.

Plus précisément, nous allons expliquer la notion de pertinence et les ratios qui permettent de la mesurer. Ensuite, nous verrons qu'il y a moyen d'ordonner les éléments de la réponse à l'aide d'estimations de la pertinence.

4.1. Pertinence de la réponse

Le but de tout système de recherche d'information textuelle est de donner l'information répondant à une question, quelle qu'ait été la façon dont cette question a été formulée. La réponse, nous l'avons vu, est un ensemble de passages de texte[1]. La qualité du système sera déterminée, en grande partie, par la qualité des réponses.

La qualité des réponses, leur **pertinence**, est évaluée en confrontant au besoin de l'utilisateur les unités d'information données par le système[2]. Nous pouvons souligner dès à présent que la pertinence dépendra en partie de la formulation de la question, puisque c'est du besoin informationnel originel que l'on tient compte[3].

Pour un besoin en information, chaque passage peut être classé dans une des catégories de la figure 4.1., qui représente la grille à la base de la quasi-totalité des mesures de pertinence.

---

1 Dans les systèmes hypertextes, la question n'est pas explicite; on peut considérer que la réponse est l'ensemble des noeuds visités.

2 Nous reviendrons plus loin sur la difficulté et les aléas de cette évaluation.

3 La définition "traditionnelle" (par opposition à la définition récente, progressiste donnée ici) ne tient pas compte de l'utilisateur; seul l'algorithme d'appariement entre questions et réponses était évalué (voir [SCHA90]).

Unités d'information	pertinentes	non pertinentes
dans la réponse	a	b
hors de la réponse	c	d

Figure 4.1. Grille de pertinence

Dans cette grille, a, b, c, et d sont les nombres d'unités d'information respectivement dans la réponse et pertinentes, dans la réponse mais non pertinentes, pertinentes mais n'apparaissant pas dans la réponse, et non pertinentes et hors de la réponse. Le nombre total de passages composant le document, N, est égal à a + b + c + d, tandis que la somme a + b nous donne le nombre d'unités d'information dans la réponse.

Intuitivement, nous pouvons, sans trop de risque, affirmer qu'un système idéal donnerait b = c = 0. Nous irons un peu plus loin dans l'analyse. A partir de la grille, nous définirons les mesures de pertinence suivantes, en insistant sur leur signification : le bruit, le silence, le taux de précision et le taux de rappel. Ensuite, nous détaillerons l'interaction entre taux de précision et taux de rappel. Enfin, nous parlerons de la méthode de construction de la grille.

4.1.1. Le bruit

Le bruit est l'ensemble des passages que le système inclut dans la réponse alors que, dans l'esprit de l'utilisateur, ils ne correspondent pas à la question. Dans la grille, le bruit est donné par la valeur b.

L'utilité de cette mesure est limitée lorsqu'il s'agit d'évaluer les performances d'un système. En effet, elle n'est pas pondérée : le bruit n'est pas aussi gênant quand a est nettement supérieur à b, et on comprend que le bruit est inévitable si le document est très grand, c'est-à-dire si le nombre d'unités d'information est élevé (b = 10 est tolérable lorsque N = 100000, et intolérable lorsque N = 20). En revanche, l'utilisateur peut directement calculer b; c'est la force de cette mesure.

4.1.2. Le silence

Le silence est l'ensemble des passages que le système n'inclut pas dans la réponse alors que, dans l'esprit de l'utilisateur, ils correspondent à la question. Dans la grille, le silence est donné par le valeur c.

Souvent associé au bruit, le silence est toutefois moins perceptible par l'utilisateur. Pourtant, il est bien plus dommageable : rares sont les circonstances dans lesquelles une réponse nettement incomplète reste satisfaisante. Comme la mesure du bruit, la mesure du silence souffre de son caractère absolu : elle ne tient pas compte du nombre d'éléments de réponse fournis. Passer sous silence une unité d'information est généralement acceptable si a = 10, mais pas si a = 1.

4.1.3. Le taux de précision

Le taux de précision ("precision" en anglais) exprime la proportion d'unités d'information pertinentes parmi l'ensemble des unités d'information de la réponse. Il est défini par :

$$P = \frac{a}{a + b}$$

Pour autant qu'il y ait au moins un élément dans la réponse, la valeur de P sera comprise dans l'intervalle [0, 1]. Une valeur proche de 1 indique que la plupart des unités d'information de la réponse sont pertinentes; une valeur proche de 0 indique que la réponse contient beaucoup de déchets.

Le taux de précision est lié à la notion de bruit; en effet, P est aussi égal à :

$$1 - \frac{b}{a + b}$$ [1]

C'est justement à cause de cette nuance que le taux de précision est très souvent préféré au bruit pour évaluer les systèmes. En outre, la valeur "normalisée" de P (au sens où elle prend toujours sa valeur dans le même intervalle) facilite les comparaisons.

---

1 On obtient ceci en substituant (a + b - b) à a dans le numérateur de P, puis en factorisant la fraction obtenue; $\frac{b}{a + b}$ est la proportion de bruit dans la réponse.

4.1.4. Le taux de rappel

Le taux de rappel ("recall" en anglais) exprime la proportion d'unités d'information pertinentes présentes dans la réponse par rapport à l'ensemble des unités d'information pertinentes pour la question. Il est défini par :

$$R = \frac{a}{a + c}$$

Quand il existe des unités d'information pertinentes (qu'elles soient dans la réponse ou non), R prendra sa valeur dans l'intervalle [0, 1]. Une valeur proche de 1 indique que la plupart des éléments de la réponse idéale apparaissent dans la réponse donnée par le système; une valeur proche de 0 indique que le système n'a donné ("rappelé") qu'une petite partie de la réponse idéale. Le taux de rappel est une mesure de la complétude de la réponse.

Le taux de rappel est au silence ce que le taux de précision est au bruit : il intègre un facteur normalisant, ce qui enrichit la mesure et rend les comparaisons plus aisées. De la même manière que le silence et le bruit sont utilisés ensemble, les taux de précision et de rappel vont de pair lorsqu'on procède à l'évaluation d'une réponse.

4.1.5. Le couple (P, R)

En pratique, c'est à l'aide du couple formé par les taux de précision et de rappel que les performances d'un système seront évaluées. En effet, ces deux mesures ont tendance à évoluer ensemble : pour obtenir une réponse complète (R proche de 1), la question devra être non discriminante, ce qui générera du bruit (la valeur de P diminue). A l'inverse, éliminer tout le bruit (P proche de 1) revient à préciser la question au maximum, ce qui aura tendance à laisser de côté des unités d'information pertinentes (cette fois, R se rapproche de 0).

Les valeurs de P et R peuvent être portées sur un graphe[1]. La courbe obtenue à l'aide du nuage de points sert à analyser, pour un même système, les sacrifices à consentir pour améliorer chacune des deux variables. Une autre utilisation consiste à tracer une courbe pour chaque système étudié et à déduire quelle est la meilleure solution.

---

1 Chaque couple de valeurs (P, R) donne un point et correspond à une stratégie de recherche, à une requête.

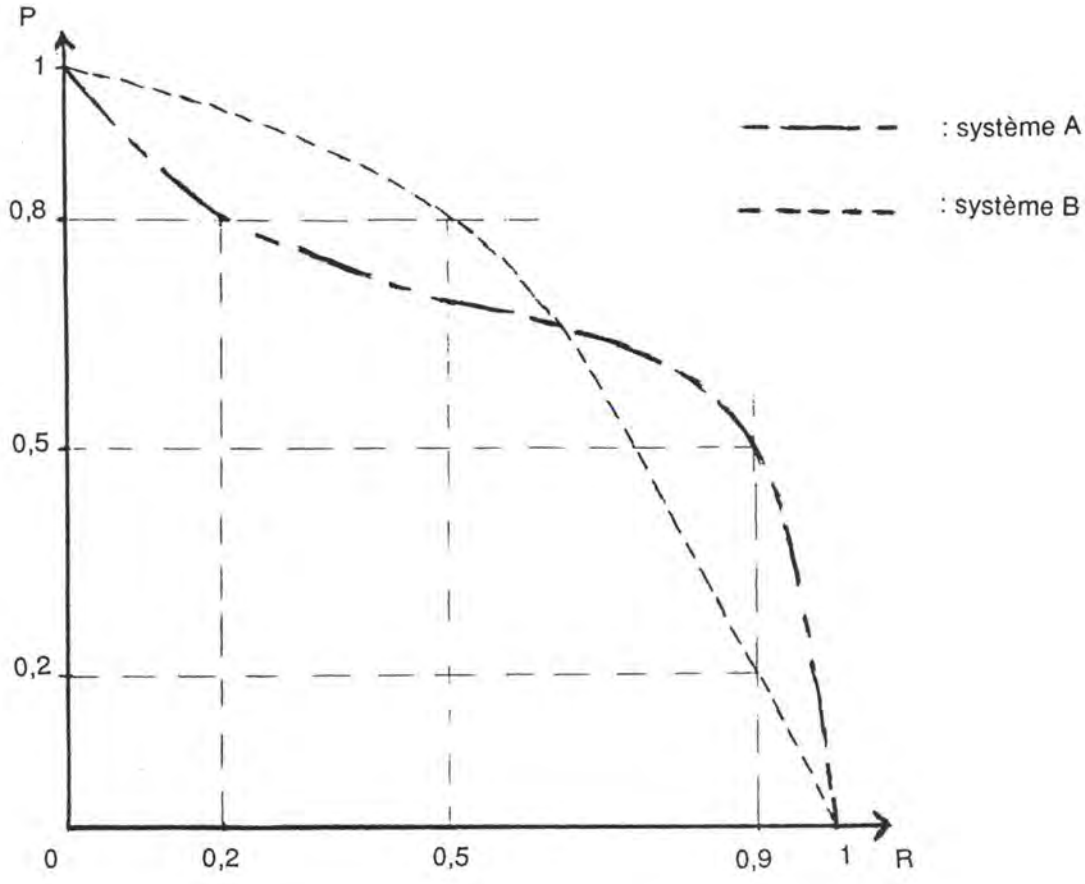

Figure 4.2. Les courbes Precision / Recall des systèmes A et B

La figure 4.2. montre un exemple de deux graphes P / R superposés; on y voit que le système A donne des réponses complètes (R > 0,9) lorsque le taux de précision est inférieur à 0,5, tandis que le système B n'arrive à la même performance qu'en abaissant le taux de précision à 0,2. Si un taux de rappel élevé est essentiel, on préférera le système A au système B.

Par contre, le système A impose de gros sacrifices en taux de rappel pour obtenir une réponse sans déchet : pour avoir, dans la réponse, moins d'un passage non pertinent sur cinq (P > 0,8), on doit accepter de passer à côté des quatre cinquièmes de la réponse idéale (R < 0,2). Le système B est donc meilleur à ce niveau-ci, car la même valeur est obtenue pour P avec R inférieur à 0,5.

4.1.6. Construction de la grille de pertinence

Pour définir les différents indicateurs de pertinence, nous avons utilisé la grille de pertinence (figure 4.1.). Nous avons dit que la

pertinence est la qualité d'une information qui apporte un élément de réponse au besoin de l'utilisateur. Nous n'avons cependant pas encore montré comment remplir les cases, c'est-à-dire comment distinguer un passage pertinent d'un autre qui ne l'est pas.

Deux méthodes sont possibles pour remplir la grille (figure 4.3.).

Méthode 1	Méthode 2
1. Cerner le besoin informationnel.	1. Cerner le besoin informationnel.
2. Utiliser le système informatique.	2. Définir la réponse idéale en examinant le document.
3. Remplir les cases a et b en examinant la réponse.	3. Faire utiliser le système informatique.
4. Remplir les cases c et d en examinant le reste du document.	4. Remplir les cases a, b, c et d en comparant la réponse donnée à la réponse idéale.

Figure 4.3. Les deux méthodes de construction

La première méthode sera utilisée pour des évaluations ponctuelles : un seul acteur, l'utilisateur, cherche à se faire une idée des performances du système. Il veille à bien déterminer ce qu'il cherche, puis il pose sa question; la réponse donnée lui permet de remplir les cases a et b de la grille, tandis que l'examen du reste du document donne les quantités c et d.

Cette méthode n'est pas utilisée lors des expériences à grande échelle et ce, pour deux raisons majeures. D'abord, elle ne fonctionne que lorsqu'il y a **un** utilisateur; dans le cas contraire, la première étape ne sera pas la même pour tous le monde, et les résultats seront faussés. Ensuite, elle ne convient qu'à de tout petits textes, sinon la quatrième étape est irréalisable.

La deuxième méthode, par contre, est largement répandue, sous des variantes diverses. Nous avons maintenant deux acteurs ou, plutôt, deux groupes d'acteurs : les **experts** et les **utilisateurs**. La première étape verra les experts construire un besoin informationnel (une question dégagée de tout formalisme). Dans un deuxième temps, ils parcourront l'entièreté du document pour définir quels sont les passages que la réponse devrait

contenir[1]; ils obtiennent ainsi deux ensembles d'unités d'information, les pertinents et les autres[2]. La troisième étape est accomplie par les utilisateurs : après leur avoir expliqué ce qu'ils doivent trouver, ils cherchent la réponse à l'aide du système informatique. Lorsqu'ils estiment avoir obtenu un résultat satisfaisant, ils en font part au superviseur de l'expérience, qui remplit la grille en comparant les unités d'information proposées par l'utilisateur à la réponse "théorique" des experts. La variante principale de cette méthode consiste à impliquer l'utilisateur dans le processus d'évaluation de pertinence (quatrième étape).

Malgré tout, cette deuxième manière de procéder n'évite pas l'évaluation exhaustive de la pertinence de **chaque** unité d'information du système (deuxième étape), qui peut, dans certains cas, se révéler économiquement irréalisable.

C'est ce qui a poussé les chercheurs à mettre au point d'autres mesures d'efficacité. Celles-ci font largement appel à des **jugements comparatifs** et à la notion d'**utilité relative** des éléments de la réponse pour construire une mesure définie à partir des seules unités d'information de la réponse (voir par exemple [FREI91] ou [SCHA90]).

4.2. Classement des éléments de la réponse

Jusqu'à présent, nous avons supposé que tous les éléments de la réponse sont égaux, présentent le même degré de pertinence. Ce n'est (malheureusement) pas le cas, et les désaccords régnant entre experts à propos de la pertinence de certains passages sont révélateurs[3].

La solution à ce problème est le classement des unités d'information par ordre de pertinence. Nous devrions plutôt dire "par ordre supposé de pertinence", car c'est le système informatique qui procède à une estimation

---

1 On aura deviné que la plus grosse difficulté se trouve cachée derrière cette phrase. En effet, les expériences ont montré ([TENO90], chap. 5) que les experts divergent considérablement lorsqu'il s'agit d'opérer la distinction pertinent / non pertinent. Pour une question et chacune des 100 unités d'information de la réponse, trois experts (seulement !) vérifient si oui ou non elles sont pertinentes; les experts ne sont d'accord que dans deux tiers des cas (oui / non confondus). Les passages litigieux sont en majorité évalués "pertinents" par deux des trois experts; c'est donc sur les passages "pas tout à fait pertinents, mais presque" que les avis divergent le plus.

2 Ces ensembles correspondent aux valeurs (a + c) et (b + d) dans la grille.

3 Lors de notre expérience, il était clair que certains paragraphes répondaient mieux que d'autres à certaines questions posées. Il est même advenu que sur un passage "douteux", l'expert se contredise : un jour il concluait (après hésitation) à la pertinence, tandis que le lendemain il trouvait le même passage non pertinent pour le même besoin informationnel.

de la pertinence. Cette estimation, basée non plus sur le besoin informationnel mais sur la question, prend souvent la forme d'une fonction de similitude, dans laquelle interviennent des paramètres comme le nombre de mots communs avec la question, leur fréquence d'occurrence dans le passage, leur fréquence d'occurrence dans l'entièreté du document, leur fréquence d'occurrence a priori (dans tout document), etc.

Un autre problème surgit maintenant : comment évaluer les taux de précision et de rappel de ces réponses ? Il faut, pour chaque sous-ensemble de la réponse, calculer ces ratios et les porter sur un graphe P / R. La figure 4.4. (inspirée de [TESK82]) illustre cette démarche. On suppose que le document comporte 100 unités d'information et que la réponse idéale (a + c) en compte 5.

- La première colonne donne les unités d'information de la réponse classées par ordre décroissant de pertinence supposée.
- La deuxième colonne indique si oui (O) ou non (N) l'unité d'information est réellement pertinente (avis de l'expert ou de l'utilisateur).
- La troisième colonne est très importante : elle donne la somme des nombres d'unités d'information dans les lignes précédentes[1]; nous aurons donc autant de couples (P, R) qu'il y a de lignes dans le tableau.
- La quatrième colonne donne le nombre d'unités d'information dans les lignes précédentes.
- Les deux dernières colonnes donnent les valeurs de P et de R pour chaque ligne[2].

---

1 Dans la réalité, chaque ligne, au lieu de correspondre à **une** unité d'information, pourra correspondre à une classe d'unités d'information dont la pertinence est supposée semblable.

2 Nous avions convenu que a + c = 5.

Unités d'information de la réponse classées	Unité d'information pertinente ?	Nombre d'unités d'information dans la réponse (a + b)	Nombre d'unités d'information pertinentes	P $(\frac{a}{a+b})$	R $(\frac{a}{a+c})$
UI_1	O	1	1	1	0,2
UI_2	O	2	2	1	0,4
UI_3	N	3	2	0,66	0,4
UI_4	O	4	3	0,75	0,6
UI_5	N	5	3	0,6	0,6
UI_6	N	6	3	0,5	0,6
UI_7	N	7	3	0,43	0,6
UI_8	O	8	4	0,5	0,8
UI_9	N	9	4	0,44	0,8
UI_{10}	N	10	4	0,4	0,8

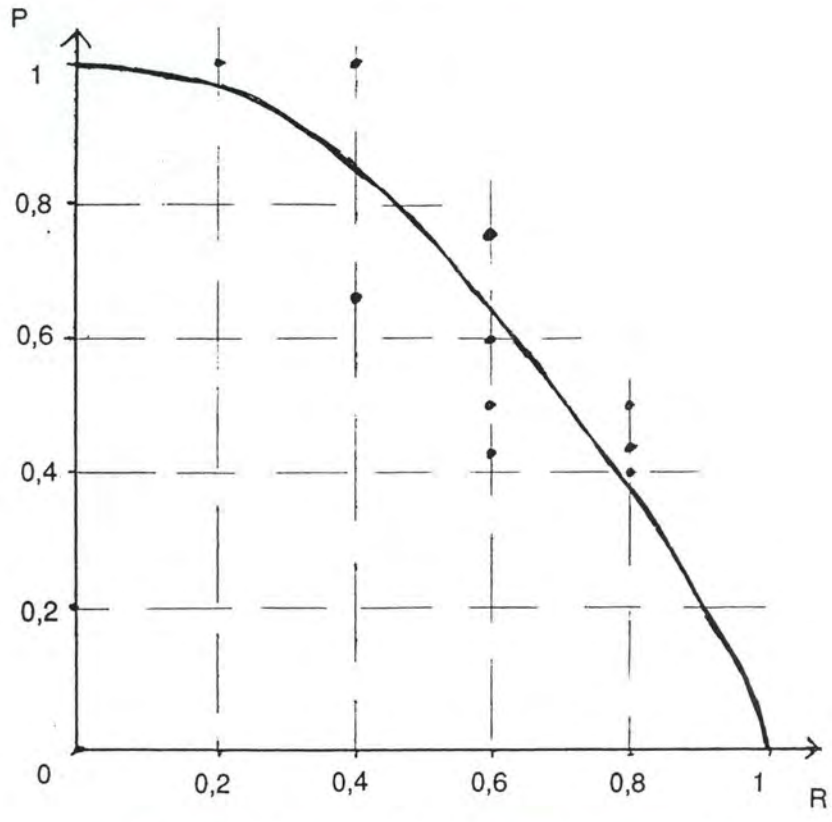

Figure 4.4. Le graphe P / R d'une réponse classée

Nous avons maintenant terminé l'étude des résultats des requêtes, ce qui marque la fin de la partie consacrée à la **recherche** d'information textuelle. Nous pouvons, avant de passer à autre chose, souligner que la pertinence est une notion essentiellement subjective, dynamique.C'est pour cela que

> *"on a renoncé à l'espoir de fournir à l'utilisateur LA réponse à LA question posée. Une réponse, c'est aujourd'hui un document*[1] *ou un ensemble de documents que l'**utilisateur** estimera pertinent(s). Cela se traduit par des systèmes qui visent moins à se substituer à lui qu'à mettre à sa disposition toute une panoplie d'outils, au premier rang desquels les moyens d'une interactivité élaborée". [DACH90], page 126.*

---

1 Il faut lire "unité d'information" à la place de "document".

Chapitre 5.
Méthodes de compression

Après avoir examiné le "moteur" de tout système de recherche d'information textuelle, c'est-à-dire les structures de données et les primitives offertes pour y accéder, nous allons nous attarder, dans ce chapitre, sur la manière de stocker efficacement le texte.

Par là, nous entendons que le texte comporte des redondances : en voyant "éch?nt?llon", vous aurez vite retrouvé le mot original, sans qu'il soit impératif d'inclure les informations redondantes que constituent les deux lettres supplémentaires "a" et "i". Les méthodes de compression de données textuelles tendent à éliminer ces redondances, afin de réduire la place nécessaire au stockage des données, sans toutefois perdre des informations ni compromettre la possibilité d'effectuer des recherches aléatoires[1].

Dans la première section, nous donnerons quelques éléments de théorie de l'information, ainsi que le modèle du prédicteur (Shannon). Dans les deuxième et troisième sections, nous expliquerons les algorithmes de Huffman et de Ziv et Lempel pour la compression de texte[2].

5.1. Notions de théorie de l'information[3]

Un **langage** (aussi appelé **source**) est défini ici comme un ensemble de N caractères.

Un **message** est une suite de M symboles puisés parmi les N caractères du langage.

---

1 Nous n'envisagerons que les méthodes de compression réversibles, qui seules permettent de reconstruire les messages initiaux exacts par une opération de décompression.

2 Des informations complémentaires sur les méthodes de compression peuvent être obtenues dans la revue IEEE transactions on information theory.

3 Les techniques de compression sont élaborées dans le cadre et avec les concepts de la théorie de l'information. Elles peuvent aisément être traduites en termes de programmes et de fichiers : le codeur et le décodeur deviennent les modules de compression et de décompression du logiciel, et le message est le fichier.

L'**entropie** d'un langage, H, mesure la quantité moyenne d'information produite par symbole de message. Si le langage est traduit en binaire de manière optimale, l'entropie est le nombre moyen de bits nécessaire à la codification d'un symbole original.

Si $X = \{x_1, x_2, x_3, \ldots, x_N\}$ est l'ensemble des N caractères de l'alphabet de départ et $p(x_i)$ est la probabilité d'occurrence du symbole x_i, on a :

$$H = - \sum_{i=1}^{N} p(x_i) \log_2 (p(x_i))$$

On peut vérifier que l'entropie est maximale quand les caractères x_i sont équiprobables.

La **redondance** d'un langage, R, mesure la différence entre l'entropie maximale et l'entropie réelle; on a :

$$R = \log_2 N - H$$

Shannon montre que l'entropie de la langue anglaise (27 symboles) est faible, et décroît si on tient compte des symboles déjà rencontrés dans le message pour prédire le symbole suivant. Prenons un exemple[1] :

```
(1)  THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG READING LAMP ON THE DESK SHED GLOW ON
(2)  ----ROO------NOT-V-----I------SM----OBL----REA----------O------D----SHED-GLO--O-
```

La ligne (1) donne l'original, et on a demandé à un sujet de prédire les symboles. Lorsque la lettre est juste, on met un tiret en regard sur la ligne (2); quand la réponse est fausse, on renseigne le sujet sur sa valeur exacte et on l'écrit sur la ligne (2).

On constate que la majorité des symboles (55 sur 81, soit près de 68 %) sont redondants, n'apportent aucune information au message, puisqu'ils sont correctement prédits. Plusieurs types de redondance coexistent dans ce texte : certains caractères sont plus fréquents a priori que d'autres, seules quelques combinaisons de mots sont permises (les mots), et au sein d'un mot, les symboles sont corrélés.

La faculté de prédiction a inspiré à Shannon un premier type de systèmes de compression, appelés systèmes à prédicteur. On peut imaginer que deux prédicteurs identiques (au sens mathématique) soient utilisés, l'un pour

---

1 Repris de [SHAN51].

coder le message, et l'autre pour le décoder, d'une façon telle que seuls les symboles ne pouvant être prédits soient effectivement transmis.

Une autre solution serait de donner, pour chaque symbole transmis, le nombre de fois qu'il a fallu deviner avant de tomber sur la lettre correcte.

Dans les deux cas, le message transmis est **réduit**, et on obtient le modèle de la figure 5.1.

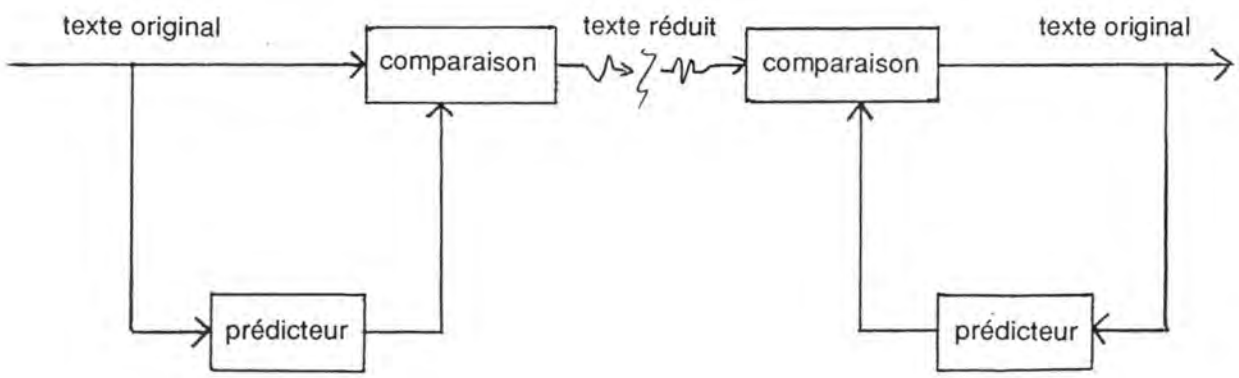

Figure 5.1. Modèle de communication utilisant le texte réduit

La conclusion de Shannon est que plus le prédicteur "connaît" le langage, plus l'entropie diminue, et plus le codage du message pourra se faire efficacement. En connaissant les 100 derniers symboles du message, Shannon calcule une entropie théorique comprise entre 0,6 et 1,3.

Ces chiffres sont valables à condition de disposer d'une connaissance parfaite de la source. Cependant, les seuls modèles qu'on ait pu se donner pour le texte sont des modèles statistiques : probabilités d'occurrence, de co-occurrence, etc. Nous sommes donc encore loin de pouvoir réduire la taille du texte à moins d'un bit par lettre (en moyenne).

Les codes utilisés pour la compression sont de plusieurs types :

- un code **fixe - fixe** associe à chaque caractère de l'alphabet de départ un et un seul caractère de l'alphabet du code. On peut enregistrer des textes dont la typographie est pauvre

(uniquement composée de majuscules et de signes de ponctuation) à l'aide de 5 bits au lieu de 8.

- un code **fixe - variable** associe à chaque caractère de l'alphabet de départ une suite de L symboles choisis parmi l'alphabet du code (L est variable). Le modèle de la source permet d'assigner aux caractères les plus redondants les codes les plus courts. Nous verrons dans la deuxième section les codes de Huffman, qui appartiennent à cette catégorie.
- un code **variable - fixe** associe à M symboles du message choisis parmi l'alphabet de départ un et un seul caractère de l'alphabet du code (M est variable). Dans ce cas-ci, les codes sont de longueur fixe, et ce sont les symboles du message qui sont groupés de manière à ce qu'ils deviennent équiprobables. Les codes de Ziv et Lempel décrits dans la troisième section sont des codes de type variable - fixe.
- un code **variable - variable** associe à M symboles du message choisis parmi l'alphabet de départ une suite de L symboles choisis parmi l'alphabet du code (M et L sont variables). Nous verrons que les codes de Ziv et Lempel ont été optimisés et débouchent sur des codes de type variable - variable.

On peut encore opérer une distinction selon le moment où la clé de codage est choisie. On a alors :

- le codage **statique**, où le codeur et le décodeur connaissent à l'avance le modèle de la source, qui ne change jamais,
- le codage **adaptatif**, où le codeur et le décodeur connaissent à l'avance le modèle de la source et les dispositions à prendre lorsque ce modèle s'avère imparfait (les codes changent en cous de compression), et
- le codage **dynamique**, où, avant de commencer à coder, le codeur transmet au décodeur la clé de décompression. A la rigueur, cette clé peut être construite et transmise (ou enregistrée) en cours de compression.

5.2. Les codes de Huffman

En 1952, D. A. Huffman publie sa méthode de construction de codes assurant une redondance minimale, dits codes optimaux[1]. Le but de Huffman est de donner un algorithme permettant de construire des codes tels que la **longueur moyenne des messages** soit réduite à son minimum.

Les contraintes qu'il impose sont les suivantes :

- deux messages différents sont codés de manières différentes, et
- lors du décodage, une fois connu le point de départ du message, aucune information supplémentaire ne sera nécessaire pour déterminer où commence et où finit tout symbole[2].

En toute généralité, les codes obtenus sont de longueurs différentes, et assureront que l'entropie[3] sera minimale. Nous donnerons les explications pour construire des codes binaires; on peut facilement généraliser pour des codes en base 3 ou supérieure.

Soient a_1, a_2, a_3, a_4, a_5 et a_6 les N (N = 6) caractères à coder de manière optimale. Leurs probabilités d'occurrence sont :

$$p(a_1) = p_1 = 0,3$$

$$p(a_2) = p_2 = 0,2$$

$$p(a_3) = p_3 = 0,15$$

$$p(a_4) = p_4 = 0,15$$

$$p(a_5) = p_5 = 0,15$$

$$p(a_6) = p_6 = 0,05$$

---

1 Cf. [HUFF52].

2 Cette condition équivaut à dire qu'aucun code ne peut être préfixe d'un autre. Un code ayant cette propriété est dit irréductible.

3 L'entropie exprime aussi la longueur moyenne des messages, en bits.

Il faut minimiser

$$L_{moy} = - \sum_{i=1}^{6} p_i \, L_i$$

où L_i sera la longueur en bits de chacun des codes.

L'entropie de ce langage donne sa limite théorique à la longueur moyenne des codes, et est égale à :

$$H = - \sum_{i=1}^{6} p_i \log_2 (p_i)$$

$$= 0,3 * 1,7370 + 0,2 * 2,3219 + 0,15 * 2,7370 * 3 + 0,05 * 4,3219$$

$$= 2,4332.$$

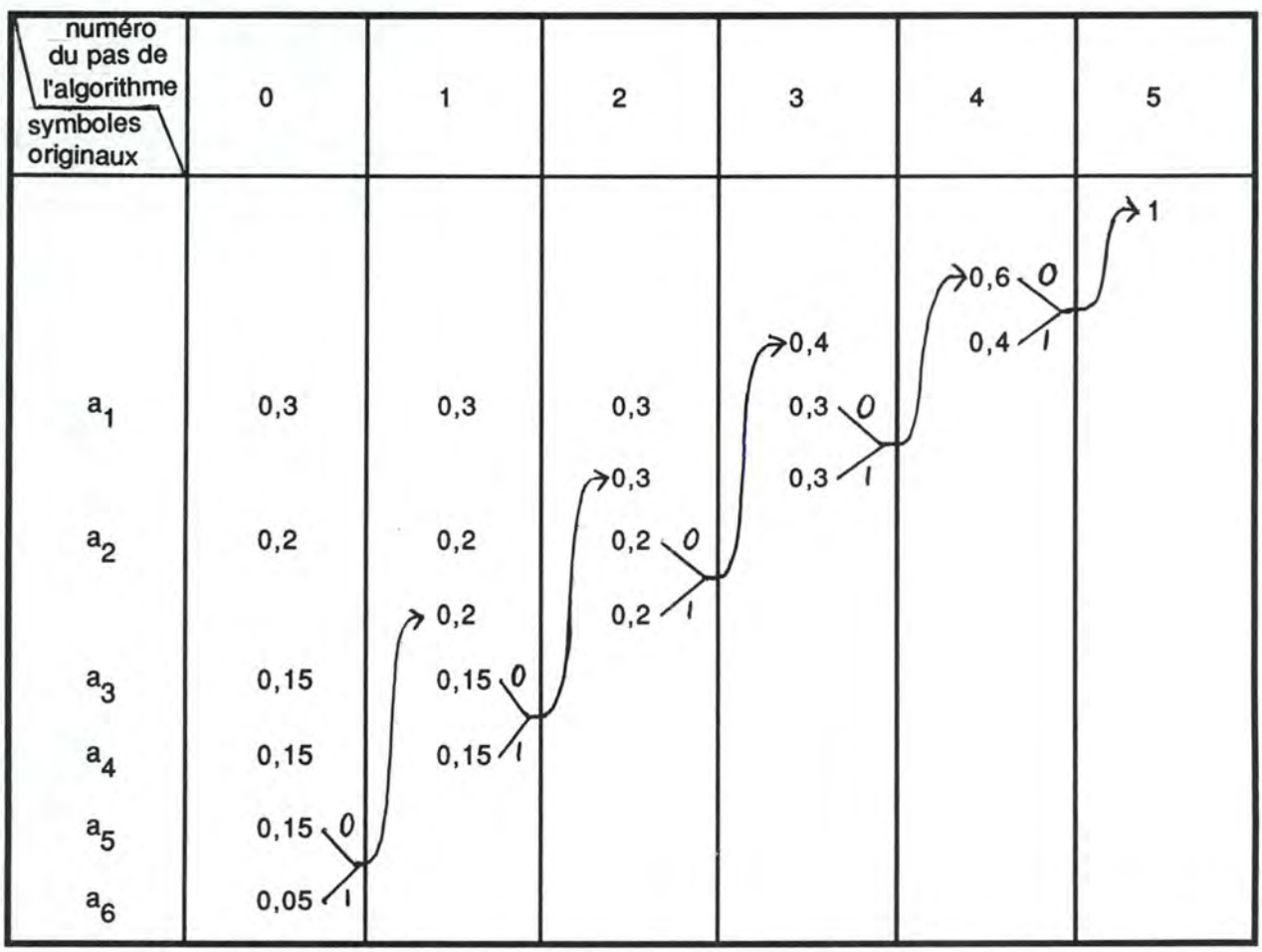

Figure 5.2. Evolution des probabilités lors de l'exécution de l'algorithme

L'algorithme est donné ci-dessous[1]; son exécution est illustrée avec notre exemple sur la figure 5.2., et son résultat fait l'objet de la figure 5.3.

1. Soit **L** la liste des probabilités des symboles originaux, correspondant aux feuilles de l'arbre[2].
2. Prendre les deux probabilités les plus petites dans L, générer un sommet parent commun aux deux sommets associés, et étiqueter l'arc du parent vers le premier descendant avec un (1), et l'arc du parent vers le second descendant avec zéro (0).
3. Remplacer les deux probabilités choisies dans L par leur somme, et associer cette somme avec le nouveau sommet intermédiaire.
4. Si la nouvelle liste **L** ne contient plus qu'un élément, arrêter; sinon, reprendre au deuxième pas.

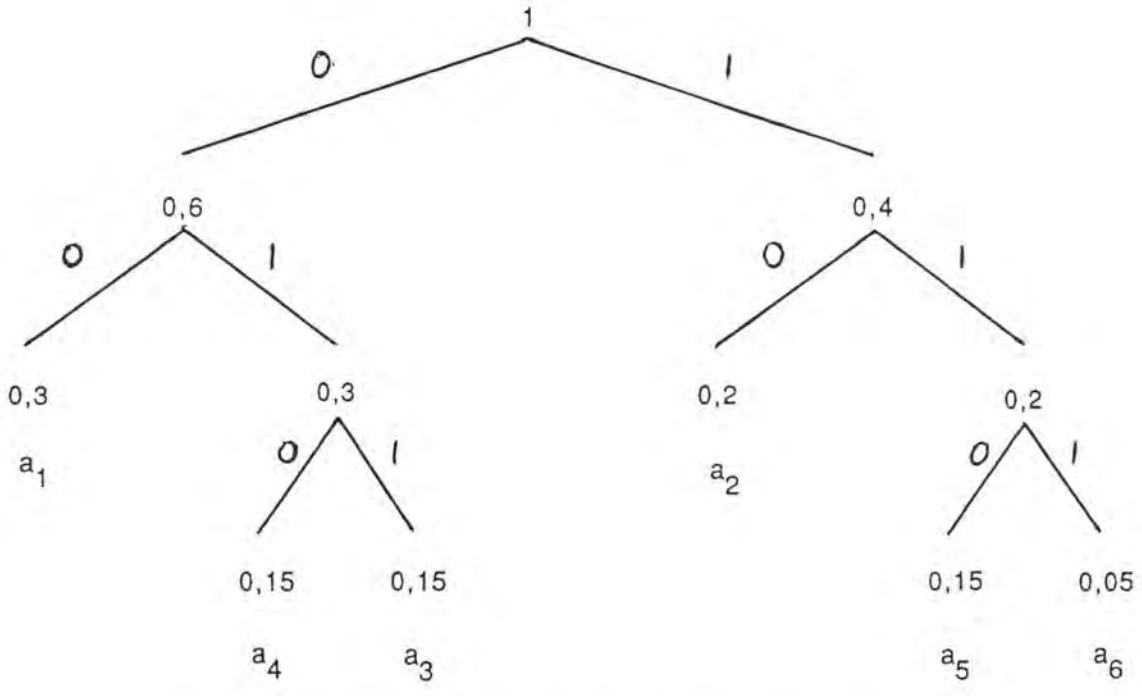

Figure 5.3. Arbre des codes générés par l'algorithme

Nous voyons qu'aux symboles les plus fréquents sont assignés les codes les plus courts. Le tableau de la figure 5.4. nous permet de calculer la longueur moyenne du code d'un symbole. Rappelons que l'entropie était de

---

1 Nous en donnons la version ascendante, proposée par Gallager ([GALL78]).

2 L'arbre n'est pas encore construit à ce stade.

2,4332; le résultat obtenu par l'algorithme est déjà très bon : on ne perd, en moyenne, que 0,07 bits par symbole codé.

a_i	p_i	L_i	$p_i L_i$	code
a_1	0,3	2	0,6	00
a_2	0,2	2	0,4	10
a_3	0,15	3	0,45	010
a_4	0,15	3	0,45	011
a_5	0,15	3	0,45	110
a_6	0,05	3	0,15	111
			$L_{moy} = 2,5$	

Figure 5.4. Les résultats de l'algorithme : la longueur moyenne des symboles

Des **variantes** de cet algorithme ont été proposées; [GALL78] donne une version adaptative, et il est possible de réaliser très facilement une version dynamique[1]. Les codes de Huffman sont parfois intégrés au sein de méthodes plus complexes; c'est le cas décrit par [SKAB91].

5.3. Les codes de Lempel et Ziv

La première version de la méthode de compression par les codes de Lempel et Ziv est apparue en 1977[2]. Le principe est complètement différent de celui des codes de Huffman; il s'agit d'un codage variable - fixe.

Ce qui est transmis (ou, dans notre domaine, enregistré sur disque), ce sont les coordonnées relatives à la position courante où a été rencontrée une chaîne de symboles similaire à celle qui commence à la position courante. A chaque étape, on transmet donc un triplet (i, j, car), où

- i donne le nombre de caractères séparant la position courante de la chaîne similaire déjà rencontrée,

---

1 La version dynamique des codes de Huffman comprendra deux passes : l'une pour compter les caractères, calculer leur fréquence et construire les codes, et l'autre pour écrire la table des codes utilisés et coder (puis écrire) le fichier.

2 Cf. [ZIV77]. Cette méthode sera désignée par LZ77.

- j donne le nombre de caractères dans la chaîne, et
- car donne, en code ASCII, le caractère suivant immédiatement la chaîne codée. En début de codage, car servira à transmettre les caractères non encore rencontrés, avec i = j = 0.

Afin de limiter les ressources requises pour coder ou décoder, on réduit le nombre de caractères gardés en mémoire; dans le même temps, on fixe des valeurs maximales pour i et j, ce qui permet de leur donner une taille fixe optimale.

Voyons sur un exemple[1] comment fonctionne la méthode LZ77. Soit babcabacbcabababa le message source à transmettre[2]. Prenons une "fenêtre" du message déjà codé de longueur 7, et une "fenêtre" du message à coder de longueur 4, ce qui nous donne des valeurs de i et j codables sur 3 bits.

Etape du codage	Caractères déjà codés							Caractères à coder				Codes transmis		
	7	6	5	4	3	2	1	1	2	3	4	i	j	car
1	-	-	-	-	-	-	-	b	a	b	c	0	0	b
2	-	-	-	-	-	-	b	a	b	c	a	0	0	a
3	-	-	-	-	-	b	a	b	c	a	b	2	1	c
4	-	-	-	b	a	b	c	a	b	a	c	3	2	a
5	b	a	b	c	a	b	a	c	b	c	a	4	1	b
6	b	c	a	b	a	c	b	c	a	b	a	6	4	-
7	a	c	b	c	a	b	a	b	a	b	a	2	4	-

Message comprimé (en notation symbolique) : (0,0,b) (0,0,a) (2,1,c) (3,2,a) (4,1,b) (6,4,-) (2,4,-).

Figure 5.5. Compression avec le code LZ77

La figure 5.5. donne le déroulement du processus de codage.

- A l'étape 1, aucun caractère n'a encore été rencontré, de sorte que i = j = 0 et car = "b".

---

1 Repris de [NUSS91].

2 Notons que ni le codeur ni le décodeur ne connaissent le langage de la source.

- A l'étape 2, le caractère "a" n'a pas encore été rencontré, donc i = j = 0 et car = "a".
- A l'étape 3, la plus grande chaîne ayant déjà été rencontrée est "b", donnée par i = 2 et j = 1; car donne le caractère suivant cette chaîne, c'est-à-dire "c".
- A l'étape 4, la plus grande chaîne ayant déjà été rencontrée est "ab", donnée par i = 3 et j = 2; car = "a".
- A l'étape 5, la plus grande chaîne ayant déjà été rencontrée est "c", donnée par i = 4 et j = 1; car = "b".
- A l'étape 6, la plus grande chaîne ayant déjà été rencontrée est "caba", donnée par i = 6 et j = 4; car n'est pas défini.
- A l'étape 7, la plus grande chaîne ayant déjà été rencontrée est "ba". Toutefois, une fois cette chaîne codée, on s'aperçoit que "ba" revient. On peut avantageusement coder cette particularité avec i = 2 et j = 4, car n'étant pas défini.

Les performances de la compression seront influencées par le nombre de caractères dont on tient compte (ici, 7 + 4). Des grandes valeurs élimineront la redondance à longue portée, mais nécessiteront plus de place pour coder i et j, et ralentiront le processus de codage.

Plusieurs variantes de LZ77 ont vu le jour :

- LZR code i et j à l'aide du code universel de Elias, ce qui permet des références à tous les caractères déjà codés et diminue la longueur moyenne de i et j.
- LZSS transmet soit (i, j) soit (car), en gardant les modifications introduites par LZR.
- LZB est une variante de LZSS où i est codé sur (roundup ($\log_2$ I) bits, où I est le nombre de caractères codés depuis le début.

Une nouvelle grande classe d'algorithmes de compression a vu le jour avec LZ78. Ici, le principe est de construire la table des chaînes déjà codées; on transmet des couples (i, car), où i est l'indice dans la table de la chaîne rencontrée, et car le caractère suivant immédiatement cette chaîne.

Une nouvelle chaîne est ensuite formée par la concaténation de la $i^{ème}$ chaine et de car, et insérée en dernière position dans la table[1].

Les variantes de LZ78 sont nombreuses; la plus connue est le code LZW (Lempel / Ziv / Welsh). La plupart de ces variantes sont analysées en détail dans des ouvrages de synthèse comme [BELL89] et [NUSS91]. Nous déconseillons la lecture des articles originaux : ils insistent plus sur la validité des algorithmes que sur leur fonctionnement et leur utilité.

---

1 La première position est toujours libre, et la transmission de son indice indique l'apparition d'un nouveau caractère.

Chapitre 6.
Interface homme / machine

Au sommet de l'architecture de tout logiciel, il y a le composant Interface[1] Homme / Machine (IHM), qui est le système informatique utilisé par une personne pour accomplir une tâche à l'aide de l'ensemble des moyens informatiques que sont les programmes de l'application. Le but de l'interface est de permettre à la personne de réaliser la tâche le mieux possible.

Les critères utilisés lors de l'évaluation d'une interface sont, pour B. Shneiderman ([SHNE87]), au nombre de cinq :

1. le temps d'apprentissage des commandes nécessaires à l'exécution d'une tâche. Ce temps d'apprentissage doit être le plus court possible.
2. la rapidité d'exécution de la tâche.
3. le taux d'erreurs effectuées par l'utilisateur, pour un temps d'apprentissage et une rapidité d'exécution donnés. Les erreurs sont soit des erreurs d'exécution (l'utilisateur se trompe dans la manipulation des commandes ou des dispositifs) soit des erreurs d'intention (l'utilisateur sélectionne une mauvaise commande). Outre la fréquence des erreurs, la facilité de correction doit être prise en compte.
4. la période de rémanence. C'est la période pendant laquelle l'utilisateur conserve la connaissance acquise, qui dépend directement de l'effort cognitif requis et indirectement du temps d'apprentissage et de la fréquence d'utilisation.
5. la satisfaction subjective à utiliser le système. La satisfaction qu'éprouve l'utilisateur s'évalue à travers le confort, l'enrichissement ressentis.

A ces cinq critères, on peut ajouter :

---

1 Pour l'utilisateur, l'interface incluera aussi les procédures d'aide et la documentation.

6. le degré de couverture des commandes. Le système informatique permet-il de réaliser toutes les activités nécessaires à l'exécution de la tâche ?

Pour interpréter correctement ces différents critères, une petite mise au point sur les concepts de tâche et d'utilisateur n'est pas superflue.

6.1. Modèle de l'utilisateur

L'utilisateur, dans le cadre des IHM, est considéré comme un "processeur humain" dont les sous-systèmes sensoriel, moteur et cognitif disposent d'une mémoire et d'un processeur. Quoique fort réducteur, ce modèle de la psychologie cognitive a permis la mise en évidence, au niveau du système cognitif, de la limite de la mémoire à court terme (de 5 à 9 mnèmes), et donc de la nécessité d'y suppléer (utilisation de termes concis et significatifs, aide à la décision telle que menus, etc.).

L'utilisateur peut être impliqué à des degrés divers dans l'univers informatique. D'un côté, une interface peut être construite sur la métaphore de la conversation, et l'utilisateur est l'opérateur qui donne ses instructions à un intermédiaire (l'interface), afin que les actions soient exécutées sur des objets non explicitement représentés. De l'autre côté, on trouve les interfaces du type mini-monde, où l'utilisateur est acteur dans le décor pourvu par l'interface; il agit sur la représentation mimétique des objets du monde réel. Il agit alors à la première personne.

La typologie des interfaces proposée par Hutchins, Hollan et Norman ([HUTC86]) fait l'objet de la figure 6.1.

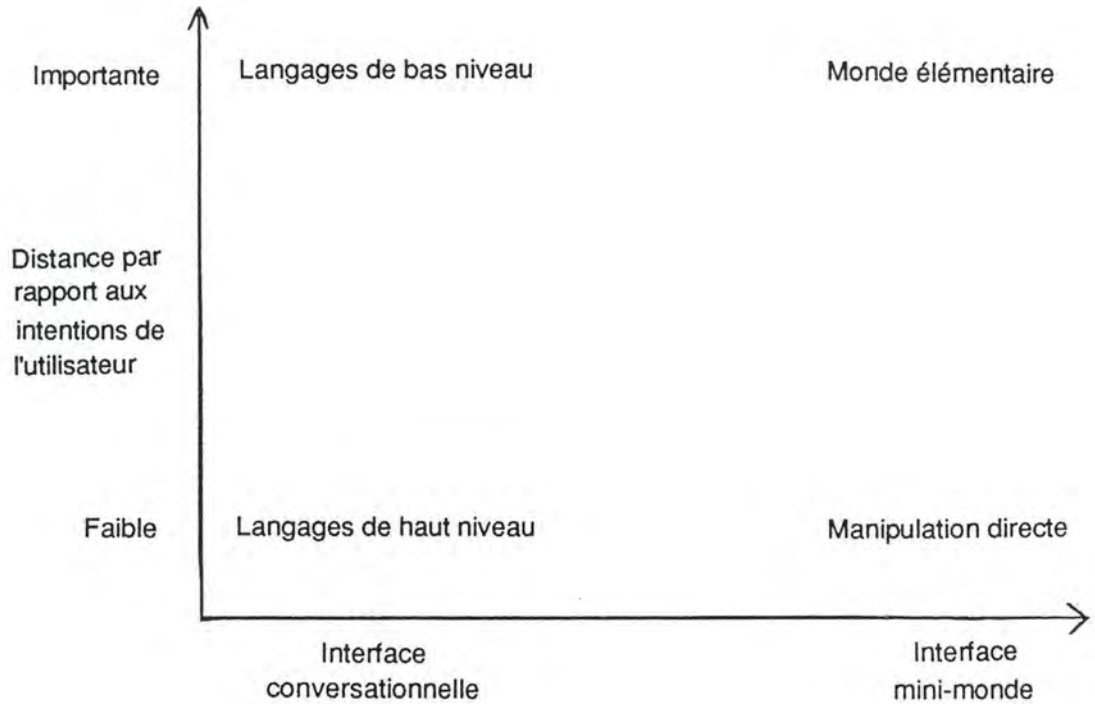

Figure 6.1. Classification des interfaces

6.2. Modèle de la tâche

La tâche, c'est le processus informatisé (ou à informatiser), en l'occurrence la recherche d'information textuelle. Un des modèles de la tâche est celui proposé par Norman ([NORM86b]) qui spécifie sept étapes; la figure 6.2. en montre le schéma.

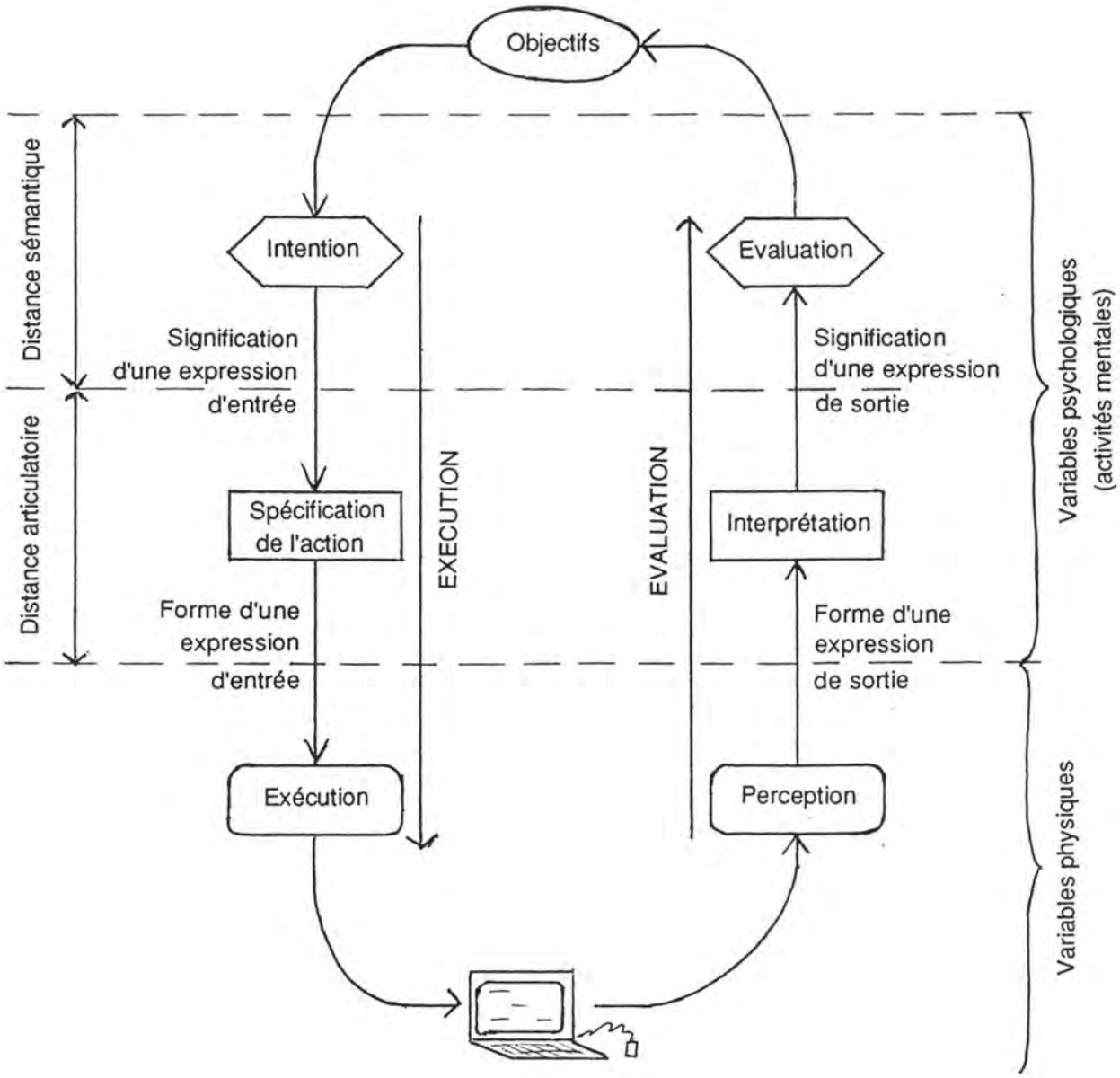

Figure 6.2. Schéma du modèle de la tâche de D. A. Norman

Le but de l'interface sera de rendre les processus d'exécution et d'évaluation plus aisés, respectivement via des mécanismes proches des représentations mentales des utilisateurs et via des dispositifs de sortie (d'affichage) dont le modèle conceptuel est facilement perçu, interprété et évalué par l'utilisateur.

On distingue ainsi la distance sémantique qui est celle existant entre les objectifs de l'utilisateur et la signification des expressions du langage de l'interface, de la distance articulatoire, qui est celle existant entre la signification des expressions et leur forme physique.

Il me semble que la recherche d'information est loin d'être une tâche triviale. En fait, je suis convaincu qu'il existe plusieurs manières d'utiliser un document, et que chaque fois la tâche est différente. On peut par exemple imaginer les cas de figure suivants :

- l'utilisateur touriste : sans autre but que de se faire une idée du système et des possibilités qu'il offre, il fait un petit tour et puis s'en va.
- l'utilisateur businessman : celui-ci sait plus ou moins ce qu'il veut, et, à défaut d'être complète, la réponse doit venir rapidement et être économiquement défendable.
- l'utilisateur fouineur : quel que soit le temps que ça prenne, il est bien décidé à retrouver le passage le plus intéressant.
- l'utilisateur méticuleux : grâce aux notes qu'il a consignées dans son agenda, il a tous les renseignement nécessaires et suffisants pour trouver ce qu'il cherche en un minimum de temps.

Cette diversité de modes d'utilisation qui, tous, doivent être possibles, participe à la complexité de la conception des systèmes de recherche textuelle. En particulier, certains utilisateurs ne savent pas vraiment ce qu'ils veulent, c'est à dire que si on prend le modèle de Norman, on a déjà beaucoup de difficultés à trouver une définition des buts.

6.3. Règles d'or

Afin de donner des lignes directrices aux concepteurs, les auteurs (Shneiderman, Scapin, Coutaz, ...) ont formulé les règles qui suivent; bien qu'il n'existe pas de recette miraculeuse, elles permettent l'obtention de bonnes interfaces dans la majorité des cas. Ces règles ne sont évidemment pas particulières au domaine de la GED. Elles sont valables pour tout système informatique.

1. Lutter pour la cohérence, éliminer les contradictions et exceptions à tous les niveaux (intra- et inter-application).
2. Offrir des raccourcis aux utilisateurs fréquents, ce qui a pour but de laisser l'utilisateur choisir lui-même le rythme de l'interaction.
3. Donner des informations en retour, de manière à ce que l'utilisateur perçoive le résultat de chacune de ses actions, et puisse évaluer son travail.

4. Construire le dialogue sur base de séquences d'actions ayant un début et une fin bien indiquées, et, plus généralement, veiller à la structuration de la tâche. Tant le stress que la quantité d'informations à conserver en mémoire s'en trouvent réduits.
5. Permettre la détection rapide et la correction aisée des erreurs ou, mieux, prévenir les erreurs.
6. Réduire la charge informationnelle à court terme.
7. Assurer la flexibilité et l'adaptabilité des interfaces aux hommes, et non le contraire.

Deuxième Partie

Comparaison de deux produits : Views et Spirit

Chapitre 7.
Présentation du livre de Kent

Nous venons de terminer la partie théorique du mémoire. Nous allons maintenant examiner comment les concepts et méthodes que nous avons étudiés sont mis en œuvre dans la pratique. Un document, le livre "Lectures on Homœopathic Philosophy", de Kent, a été informatisé; nous en tirerons les enseignements dans la deuxième partie de ce travail.

Ce chapitre, introduit l'expérience, en expliquant dans quatre sections :

- le contenu du livre,
- la démarche de l'expérience,
- la découpe du livre en unités d'information, et
- le choix des conventions d'encodage.

7.1. Le contenu du livre

Paru en 1900 sous le titre "Lectures on Homœopathic Philosophy", le livre de J. T. Kent, rédigé en anglais, reprend le texte des conférences qu'il a données à la Post-Graduate School of Homœopathics (Philadelphie). L'édition à partir de laquelle les versions électroniques ont été réalisées est celle de B. Jain Publishers (New Delhi).

Pour comprendre ce livre de Kent, il faut d'abord en introduire un autre, l'Organon, de Hahnemann. L'Organon est un manuel, écrit originellement en allemand, composé de 291 paragraphes numérotés, énonçant les principes de base sur lesquels repose la démarche homéopathique. Hahnemann en a réalisé six éditions, la sixième étant la plus achevée et comprenant maints remarques et commentaires. Une version existe en anglais, dont la traduction a été assurée par Boericke, un troisième grand nom de l'homéopathie.

Les "Lectures on Homœopathic Philosophy" ont été écrites dans un but pédagogique : chacune des conférences introduit le lecteur à un passage de l'Organon. Comme le précise Kent dans sa préface, certains paragraphes de l'Organon sont explicites et ne nécessitent aucun support à l'apprentissage,

alors que d'autres, plus obscurs ou plus complexes, doivent faire l'objet d'éclaircissements; c'est l'objet du livre.

7.2. La démarche de l'expérience

Le but de l'expérience est d'**illustrer** par un exemple les notions vues dans la partie théorique du mémoire. Dans ce cadre, il était important de choisir deux logiciels assez différents l'un de l'autre, de manière à couvrir une grande partie du domaine étudié, et assez proches que pour permettre les comparaisons.

Views, de Folio Corporation, était déjà à l'étude au sein de la société Archimède, ce qui nous a amené à le retenir en priorité. Spirit s'est vite révélé être le complément idéal : ses approches et solutions sont diamétralement opposées, et permettent de mettre en lumière la plupart des éléments de théorie que Folio n'utilise pas.

Dans un premier temps, il a fallu préparer le texte encodé pour le rendre utilisable par Views et Spirit. Ce travail de base est décrit dans les sections qui suivent.

Ensuite, à l'aide du texte ainsi préparé, nous avons construit deux versions du document, l'une sous Views et l'autre sous Spirit. A cette occasion, nous avons découvert la manière dont fonctionne chacun des systèmes; les huitième et neuvième chapitres reprennent l'essentiel de nos observations.

Enfin, la dernière partie de l'expérience consistait à confronter les utilisateurs aux deux systèmes, et à leur faire évaluer, avec leurs critères, les solutions proposées. Une mini-étude de pertinence nous a permis d'approfondir un peu la comparaison. Les résultats de cette étude, ainsi que l'avis d'un utilisateur et du "concepteur", font l'objet du dixième chapitre[1].

7.3. La découpe en unités d'information

7.3.1. La structure du livre

Pour parvenir à une découpe convenable du livre, il faut d'abord en extraire la structure, et identifier dans cette structure quel sont les "ato-

---

1 Pour des raisons indépendantes de notre volonté, le dixième chapitre n'a pas pris l'ampleur que nous voulions lui attribuer : plus que les résultats, c'est la démarche qui importe.

mes". Une fois cette opération accomplie, le dialogue avec un futur utilisateur pourra déterminer quel est le meilleur choix pour l'unité d'information.

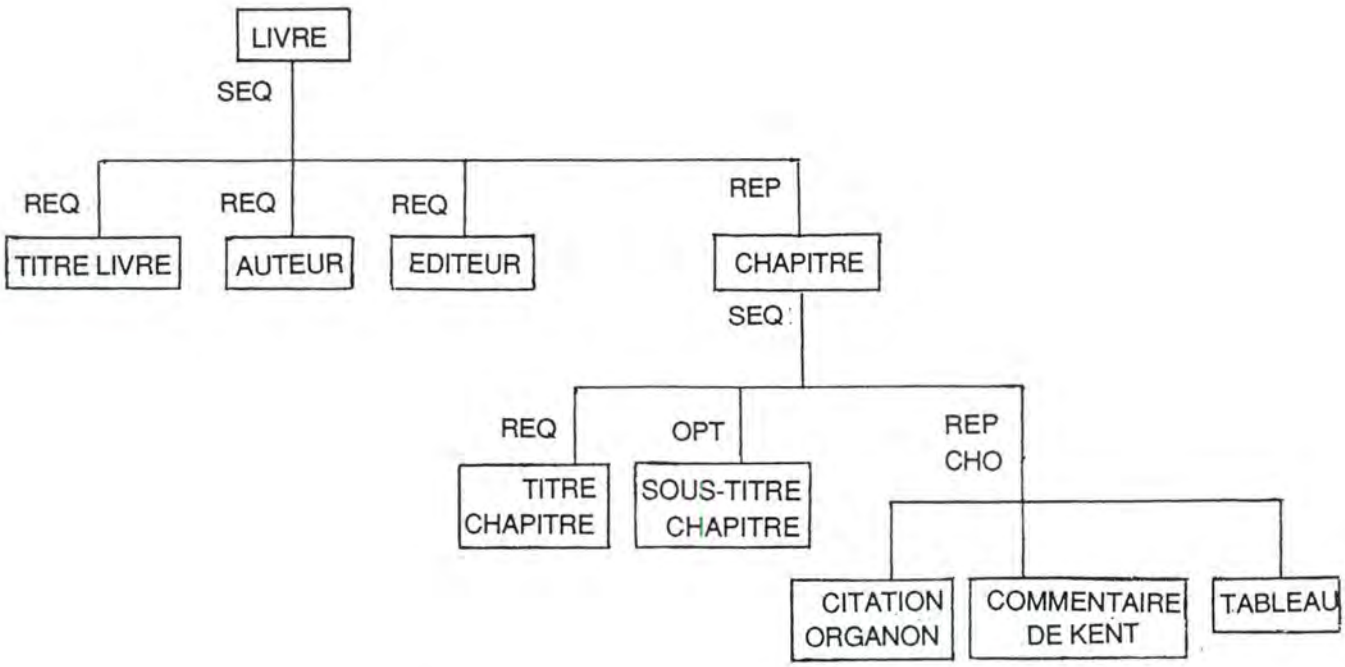

Figure 7.1. Structure générique logique (formalisme ODA)

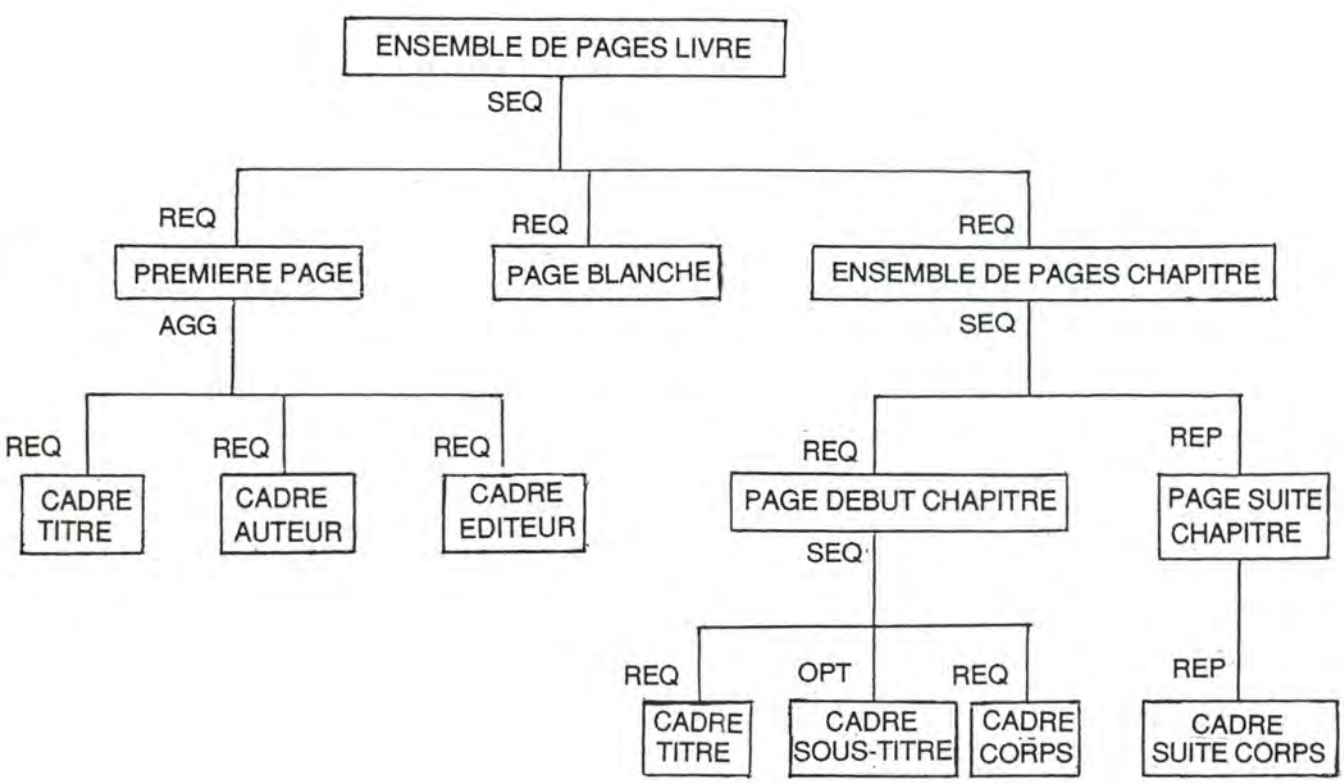

Figure 7.2. Structure générique physique (formalisme ODA)

Les figures 7.1. et 7.2. donnent les structures génériques logique et physique du livre de Kent, en adoptant le formalisme ODA décrit au deuxième chapitre. La description complète n'est pas donnée : des éléments complexes, tels le péritexte, l'entête des pages, la numérotation, etc. ont été volontairement omis, afin de ne pas surcharger les schémas.

Les structures génériques des deux figures sont la forme générale (et concise) des structures spécifiques associées. Ces structures spécifiques (la figure 2.1. en était un exemple) sont déductibles des structures génériques données.

7.3.2. Les unités d'information

La structure logique étant la plus riche sémantiquement, l'utilisateur préférera souvent cette découpe à la découpe associée à la structure physique (pages ou blocs de texte). Ici, c'est donc le **paragraphe** qui a été retenu comme unité d'information.

Les informations complémentaires à prendre en compte sont surtout liées à la structure : un passage appartient à un chapitre (identifié par son titre et son sous-titre) et peut contenir soit une citation de l'Organon soit un commentaire de Kent. Les informations propres au livre comme les numéros de page peuvent être conservées dans la version informatisée, mais elles perdent leur sens, et, surtout, ne sont valables que pour une édition particulière.

7.4. L'encodage

7.4.1. Les besoins

Les besoins au niveau de l'encodage dépendent directement de la nature des informations qu'on désire inclure dans le système. Dans notre exemple, nous voulions être à même d'identifier :

- le texte d'une unité d'information,
- son type (commentaire ou Organon),
- les titre et sous-titre du chapitre y associés,
- les mots du texte en italiques, et
- les mots du texte formant un tableau.

7.4.2. Les premières conventions

L'encodage du texte a été réalisé par le personnel de l'ASBL Archimède. Je pense qu'il faut souligner deux choses. D'une part, le document original (sur papier) est de qualité médiocre. En effet, il y apparaît de nombreuses erreurs de typographie (citations jamais terminées, parenthèses manquantes, table des matières incohérente, index inexistant, …), et l'impression a été négligée (lignes dansantes, taches, etc.). D'autre part, les encodeuses ne comprenant pas ce qu'elles tapent, des erreurs supplémentaires sont venues s'ajouter à l'anglais parfois désuet du début du siècle.

Le poste de travail utilisé à l'encodage est un terminal du VAX, doté d'un éditeur rudimentaire. Ce dernier n'assurant pas le "word wrapping" aujourd'hui disponible dans tout traitement de texte sur PC, les encodeuses insèrent un retour ligne[1] à la fin de chaque ligne **écran**. En outre, un retour ligne est inséré à la fin de chaque phrase. Les conventions adoptées font l'objet de la figure 7.3.

retour ligne	fin de ligne écran
	fin de phrase
	fin de titre
	ligne vide
tabulation	début de paragraphe
	alignement de données de tableau
#R	marque le début de la ligne où commence le titre du chapitre
#D	marque le début de la ligne où commence le sous-titre du chapitre
#P	marque le début d'une ligne où figure un numéro de page (seuls ce #P et le numéro de page apparaissent sur la ligne, séparés par un espace)
<	marque le début d'un passage en italiques
>	marque la fin d'un passage en italiques

Figure 7.3. Les conventions originales

---

1 "Retour ligne" est utilisé à la place de l'expression, plus précise mais plus lourde, de "retour chariot et passage à la ligne suivante" (carriage return / linefeed).

7.4.3. Les défauts des premières conventions

Mises à part les erreurs linguistiques (archaïsmes), typographiques et de dactylographie ayant déjà été relevées, le matériau présente quelques défauts liés au choix des conventions. On peut distinguer d'une part l'ambiguïté de certains codes et d'autre part la perte d'information par rapport au document original.

a) Ambiguïté

Les multiples utilisations du **retour ligne** compliquent son interprétation (sans pour autant le rendre ambigu). Selon les caractères qui précèdent et qui suivent, on pourra déterminer la "valeur" d'un retour ligne particulier; en aucun cas il ne sera possible de dégager cette valeur à partir des seuls codes le composant (ASCII 13 et 10).

La **tabulation**, quant à elle, est utilisée après un retour ligne pour indiquer un début de paragraphe. Son ambiguïté vient du fait qu'elle est aussi utilisée (n'importe où sur la ligne) pour aligner des données dans un tableau[1]. Nous verrons ultérieurement qu'on peut maintenir les deux usages si l'on introduit des conventions délimitant les tableaux, inexistantes à ce stade.

Cependant, ce sont certainement les **signes "<" et ">"** qui sont les plus ambigus : ils apparaissent indifféremment, à tout endroit dans le texte, comme marqueurs et sous leur valeur propre. Ici, il n'existe aucune règle syntaxique permettant de lever l'ambiguïté. Il est donc impératif d'introduire des marqueurs non ambigus pour indiquer le début et la fin d'un passage en italiques. "<DI>" et "<FI>" seront choisis, essentiellement pour leur qualité mnémotechnique.

b) Informations manquantes

L'édition indienne du livre est assez "pauvre", mais elle comporte néanmoins des informations donnant des clés de lecture. On peut relever, outre la table des matières et les numéros de pages, le changement de typographie pour les citations des paragraphes de l'Organon et l'utilisation des italiques pour mettre certains mots en évidence.

---

1 Notons dès à présent qu'il s'agit là de la seule utilisation alternative : toutes les tabulations en dehors des tableaux sont en début de ligne et marquent le début d'un nouveau paragraphe.

On peut aussi inclure dans les informations manquantes le **caractère tabulaire** de quelques lignes de texte des pages 175 et 229. En effet, s'ils constituent des unités bien définies dans le livre, ces tableaux sont noyés au sein du flux des caractères ASCII; en aucune façon ils ne restent identifiables.

7.4.4. Les secondes conventions

Le nouvel ensemble de conventions doit permettre d'indiquer : les passages en italiques, les paragraphes de l'Organon, les unités de texte telles que les tableaux et, éventuellement, les citations non accompagnées d'un changement de typographie. Il faut évidemment maintenir les conventions de titre et de sous-titre[1].

Ces dernières sont conservées intactes; on postule que la séquence de codes ASCII 13, 10, 35 (c'est-à-dire un signe "#" en début de ligne) ne peut jamais faire partie du texte initial et est donc non ambiguë. On garde donc #R, #D et #P.

Les autres délimiteurs concernent ainsi les débuts et fins d'italiques, de passages Organon, de tableaux et de citations. Les marqueurs <DI>, <FI>, <DO>, <FO>, <DT>, <FT>, <DC> et <FC> sont construits en prenant les signes "<" et ">" pour encadrer les codes de deux lettres, la première indiquant s'il s'agit du début ("D") ou de la fin ("F") du texte sur lequel porte le code, et la deuxième de quelle nature est le texte encadré ("I" pour italique, "O" pour Organon, "T" pour tableau et "C" pour citation). Par exemple, "Bonjour, <DC><DI>coucou<FI> et au revoir<FC>" indique que "coucou et au revoir" est une citation dans laquelle "coucou" est en italiques[2].

Tous les marqueurs construits dans ce symbolisme commencent soit par "<D" soit par "<F". Pour assurer la non ambiguïté, il faut offrir une règle permettant d'évaluer si ces deux couples de caractères sont à prendre comme éléments de texte ou éléments de marqueurs. Arbitrairement, posons que, pour qu'ils soient pris **tels quels**, "<D" et "<F" seront encodés "<<D" et "<<F". Le fait d'inclure D et F dans la suite de caractères à exception réduit évidemment à très peu les endroits où le redoublement du signe "<" sera exigé. De cette manière, on diminue les erreurs d'encodage, tout en intro-

---

1 Les numéros de page peuvent être maintenus; ils seront ignorés par les logiciels lors de l'input si on ne veut pas en tenir compte.

2 Cette manière de construire les marqueurs est inspirée de SGML.

duisant la non ambiguïté des conventions. La figure 7.4. reprend le détail des nouvelles conventions.

retour ligne	fin de ligne écran
	fin de phrase
	fin de titre
	ligne vide
tabulation	début de paragraphe
#R	marque le début de la ligne où commence le titre du chapitre
#D	marque le début de la ligne où commence le sous-titre du chapitre
#P	marque le début d'une ligne où figure un numéro de page (seuls ce #P et le numéro de page apparaissent sur la ligne, séparés par un espace)
<DI>	marque le début d'un passage en italiques
<FI>	marque la fin d'un passage en italiques
<DO>	marque le début d'un paragraphe passage de l'Organon
<FO>	marque la fin d'un paragraphe passage de l'Organon
<DT>	marque le début d'un tableau
<FT>	marque la fin d'un tableau
<DC>	marque le début d'une citation
<FC>	marque la fin d'une citation

Figure 7.4.. Les nouvelles conventions

Notons finalement que les codes de début et fin de tableau garantissent la non ambiguïté des tabulations. En aucun cas, en effet, une tabulation n'apparaît en dehors d'un tableau pour un autre motif que celui de séparé deux paragraphes.

Chapitre 8.
Présentation de Views

Views[1] est le premier logiciel que nous avons choisi pour illustrer la théorie. Ce chapitre est structuré en trois sections : la première explique les concepts de base que Views utilise pour modéliser les documents; la deuxième explique le cycle de vie d'un document; la troisième montre l'emploi des opérateurs booléens et positionnels pour la formulation des requêtes; la quatrième détaille les spécificités de Views, telles la visualisation de l'arbre de recherche, la compression des données, la protection des informations, etc.

8.1. Le document dans Views

Un document tel que le livre de Kent est transposé au sein de Views sous la forme d'une infobase[2].

> *"Une infobase est un recueil d'informations contenant des folios, des références, des liens et des groupes stockés dans un seul fichier". [FOLI91]*

Nous allons revenir plus systématiquement sur chacun des types d'objets constituant l'infobase.

8.1.1. Les folios

Dans Views, l'unité d'information est appelée **folio**; ce sont les mêmes règles régissant la découpe du document en unités d'information qui seront utilisées pour structurer l'infobase en folios. La réponse à une question sera donc un ensemble de folios.

Le manuel de Views recommande la création de petits folios, c'est pourquoi le paragraphe de texte sera choisi dans la plupart des cas.

---

1 Views est une marque déposée de Folio Corporation. Notre propos vaudra pour la version 2.0a Auteur en français de Views. Les versions Runtime et Personal Edition offrent moins de possibilités à l'utilisateur.

2 Les modalités de la conversion sont expliquées dans la troisième section du chapitre.

Un folio est constitué de son contenu textuel, ainsi que d'une **référence** (optionnelle). La référence d'un folio est une sorte de titre; elle est affichée à l'écran lorsqu'on consulte le folio. C'est un point de repère pour l'utilisateur.

8.1.2. Les groupes

Les groupes nous offrent un premier moyen de structurer l'infobase. Un **groupe** est une collection de folios. Un groupe peut contenir plusieurs folios, et un folio peut appartenir à plusieurs groupes.

Dans notre expérience, chacun des paragraphes a fait l'objet d'un folio. Pour reproduire la structure du livre, nous avons créé un groupe par chapitre, et inséré dans ces groupes les folios correspondants. Nous pouvons imaginer d'autres groupes : les citations de l'Organon, par exemple, pourraient être collectées au sein d'un groupe.

Pour autant qu'il en ait le droit, l'utilisateur peut, à tout moment, créer et supprimer des groupes, ainsi qu'y adjoindre des folios, ou en retirer. Il pourra ainsi personnaliser le document, et lui donner une structure plus proche de la façon dont il le perçoit.

8.1.3. Les liens

Au delà de la structure statique donnée par les groupes, on désirera souvent inclure dans l'infobase une structure dynamique. Les liens de Views sont similaires aux liens hypertextes : ils permettent un parcours différent du document, sans passer par la recherche[1].

Deux grands types de liens sont à distinguer dans Views :

- les liens internes ou hypertextes, représentés par un triangle sur pointe, ont comme référé un point d'une infobase (quelconque). Celui-ci sera, selon les cas, un folio, un groupe, un ensemble de folios définis par une requête, ou même un point quelconque d'une autre infobase.
- les liens externes ou liens programmes, représentés par une flèche à double sens, permettent de définir des points de

---

1 La différence avec les liens hypertextes est que le lien de Views est un lien aller / retour : une fois que l'utilisateur a traversé le lien, il ne se trouve plus dans le même espace de travail; une nouvelle **fenêtre**, une nouvelle **vue** ont été créées. Pour revenir à son point de départ, il ne parcourra pas le lien en sens inverse : il fermera la fenêtre.

sortie vers l'extérieur de Views. Utiles pour visualiser des images ou pour reproduire des sons, les liens programmes seront aussi utilisés pour mener l'utilisateur vers n'importe quel autre utilitaire (traitement de texte, application connexe, etc.).

Nous avons créé une table des matières où chaque chapitre est accompagné d'un lien hypertexte menant vers les folios du groupe[1].

8.1.4. La page de garde

Le dernier composant d'une infobase (mis à part l'index[2]), c'est la page de garde. Très utile, elle permet d'inclure dans le fichier des caractéristiques globales ne faisant pas partie du texte, comme le nom de l'auteur, la date de création, le copyright, etc.

La page de garde est créée indépendamment de l'infobase, à l'aide d'un outil spécifique (TITLEPG.EXE); elle est affichée lorsque l'utilisateur s'apprête à entrer dans l'infobase, c'est-à-dire lorsque le nom de l'infobase est en surbrillance.

8.2. Le cycle de vie d'une infobase[3]

8.2.1. La conception

La conception de l'infobase est essentielle; il faut discuter, si possible avec les futurs utilisateurs, de leurs attentes vis-à-vis de la version informatique du document. Pour cela, il faut connaître les habitudes d'utilisation : quelles sont les unités d'information, les recherches couramment effectuées, etc.

Le but de la conception est de dégager :

- le critère de découpe en folios,
- l'organisation primaire (séquentielle) des folios,
- les groupes à définir pour l'ensemble des utilisateurs, et

---

1 Un groupe de folios a été défini par chapitre.

2 L'infobase comprend d'autres éléments que nous ne détaillerons pas, comme la "bannière" et le titre; nous expliquerons plus loin les droits d'accès, aussi enregistrés dans l'infobase.

3 Cette section décrit le cycle de vie des infobases destinées à être diffusées, voire commercialisées.

- les liens à inclure dans l'infobase.

8.2.2. La production expérimentale

Avant de procéder à la production proprement dite, il est conseillé de passer par un prototype de taille réduite. On a ainsi l'occasion de se rendre compte des difficultés, des défauts de conception, etc.

Chaque phase de la création devra être testée : le filtrage, la segmentation, le référencement, le groupage et la création de liens. Dans notre expérience, c'est à ce stade-ci que nous avons découvert les carences du fichier original (codes de formattage mal définis).

8.2.3. La production de l'infobase

Deux méthodes sont possibles : soit on procède au coup par coup, comme dans la phase expérimentale, en créant l'infobase la plus simple et en l'améliorant par modifications successives à l'intérieur de Views, soit on crée une chaîne de programmes assurant la création. Nous nous limiterons à la deuxième approche, seule valable dans un environnement commercial[1].

Outre Views, Folio fournit une série d'outils logiciels destinés à la production d'infobases. Les plus importants sont :

- CREATE permet de passer d'un fichier texte ASCII à une infobase; il crée l'index, interprète les définitions de folios, de groupes, de liens, etc. Il compacte le fichier final.
- VE est l'éditeur de Views et sert à définir les arguments pour les autres utilitaires. Dans la phase expérimentale, il permet de mimer le comportement des utilitaires.
- FSR (acronyme de Filter, Segment and Reference) est un utilitaire de recherche / remplace très évolué. Comme son nom l'indique, il intervient notamment lors du nettoyage, de la segmentation et de la création des références du fichier d'entrée.
- GRABTEXT permet d'extraire des lignes d'un fichier texte.

---

1 Dans le premier cas, il est très difficile de modifier ce qui a déjà été fait, il est quasiment impossible de documenter son travail, et la procédure de création n'est pas réutilisable.

- HIERARCH sert, entre autres, à créer la table des matières et les références.
- TITLEPG et TPGCOPY sont prévus pour créér des pages de garde et les attacher aux infobases.

Les étapes conduisant à la création de l'infobase sont au nombre de sept. Chacune d'elles consiste à définir les fichiers d'arguments nécessaires aux outils utilisés.

1. Vérifier la cohérence du texte de départ et de son balisage; retirer les codes inutiles, remplacer les codes utiles par leur équivalent Views.
2. Segmenter le texte en folios.
3. Créer les références des folios.
4. Créer la table des matières.
5. Créer les groupes.
6. Créer les liens.
7. Créer l'infobase.

La figure 8.1. détaille les étapes de la création de l'infobase de notre expérience. Les noms des fichiers d'arguments sont entre parenthèses; leur contenu fait l'objet des listings en annexe[1].

---

1 L'explication approfondie de chacun de ces traitements nous semble déplacée dans le cadre de ce mémoire; pour plus de précisions, vous pouvez contacter Jean Michalski au 067/21.62.64, ou Anne de Baenst au 081/72.49.94.

Figure 8.1. La chaîne de création de l'infobase PHILKNT.NFO

8.2.4. La mise à jour de l'infobase

Une fois distribuée, l'utilisateur pourra, à son gré, personnaliser l'infobase, en lui adjoignant des liens, des groupes, etc. Ces modifications ne sont évidemment pas récupérables lorsqu'une nouvelle version de l'infobase est mise sur le marché.

8.3. Les requêtes

La création de l'infobase comprend le processus de création de l'index. L'index de Views contient les formes rencontrées dans le texte des folios. Toutes les lettres et les chiffres sont des caractères concordables. Les caractères non concordables sont l'espace et la tabulation. Les caractères semi-concordables recouvrent, en gros, les signes de ponctuation.

Parmi les cas particuliers, on peut noter :

- l'apostrophe (concordable),
- les caractères semi-graphiques (non concordables),
- les lettres grecques (ASCII 224 à 238) (concordables), et
- les codes ASCII inférieurs à 32 (non concordables; certains sont des codes de présentation).

L'index de Views est un index dense, c'est-à-dire qu'il contient toutes les formes présentes dans le texte; il n'y a pas de mots vides.

Nous l'avons vu dans le troisième chapitre, ce sont les opérateurs booléens et positionnels qui sont généralement utilisés pour construire les requêtes sur formes. Views n'échappe pas à la règle et propose les opérateurs suivants :

- "and", représenté par l'espace ou le &,
- "or", représenté par le slash (/),
- "xor", représenté par le tilde (~),
- "not", unaire ou binaire, représenté par l'accent circonflexe (^),
- "adj", représenté par la suite des mots entre guillemets,
- "adjN", représenté par la suite des mots entre guillemets, suivie du nombre maximal de mots compris entre le premier et le dernier des termes, et

- "nearN", représenté comme "adjN", sauf qu'on intercale entre le deuxième guillemet et le nombre un a commercial (@).

Finalement, Views propose les caractères jokers :

- "?" représente n'importe quel caractère, et
- "*" représente n'importe quelle suite de caractères.

Les jokers peuvent être utilisés n'importe où[1] (en début comme en fin de mot). Les parenthèses servent à forcer l'ordre d'évaluation des opérateurs.

Voyons quelques exemples de requêtes. L'utilisateur veut obtenir les passages donnant la réponse à : "How long can the homœopathic aggravation take ?". Il introduit, en première requête[2] :

homœopathic aggravation take

Constatant que la réponse contient deux folios, dont un seul est pertinent, il retire "take" de la question (ce mot est dans le passage, mais indépendant de la phrase intéressante). Par contre, il ajoute "long", qui se trouve dans ce passage. On a alors :

homœopathic aggravation long

Ici, trois folios forment la réponse; ils sont tous trois pertinents, mais "long" n'apparaît pas là où on s'y attendait. "Short" est essayé, mais on n'obtiendra aucune information complémentaire.

8.4. Spécificités de Views

8.4.1. Première évaluation du résultat avec arbre

L'interface homme / machine de Views est particulièrement soignée, surtout pour la recherche. Pour arriver à l'écran de recherche, il suffit d'appuyer sur la barre d'espacement. Nous avons déjà vu que les opérateurs ont été "traduits" dans des représentations sémantiquement riches. En outre, l'utilisateur, lorsqu'il introduit sa question, dispose de moyens d'évaluation simples et efficaces.

---

1 Sauf à l'intérieur de guillemets, où ils sont recherchés littéralement.

2 L'espace entre les mots correspond au "et" ("and") booléen.

A gauche de l'écran, une fenêtre lui donne accès à l'index de l'infobase. Au fur et à mesure qu'il tape les caractères d'un mot, la fenêtre se positionne sur l'endroit de l'index où le mot se trouve. On peut ainsi vérifier si les jokers sont utiles ou non, si la question comporte des fautes d'orthographe, etc. La fenêtre de l'index est aussi très pratique pour sélectionner les mots de la recherche : en cliquant sur les lignes désirées, on évite de devoir tout taper au clavier.

A droite de l'écran, une autre fenêtre donne l'arbre de recherche, avec le nombre de folios que contient la réponse. L'utilisateur raffine sa question à loisir, tout en étant sûr de ne pas avoir de mauvaise surprise (réponse trop courte ou trop longue).

En reprenant le modèle de la tâche de Norman[1], on peut dire que les sigles utilisés à la place des "and", "or", etc. réduisent la distance articulatoire lors de l'exécution, alors que le fait de travailler sur les formes du texte (avec toutefois la possibilité d'insérer des jokers) plutôt que sur les mots ou les concepts maintient la distance sémantique à un niveau assez élevé.

Lors de l'évaluation, on assiste au même phénomène : la distance articulatoire est presque nulle grâce à l'arbre de recherche qui donne explicitement la signification de l'expression, ne laissant aucun doute sur son interprétation, tandis que la distance sémantique n'est réduite que par l'option "Focus"[2], qui aide à l'évaluation plus rapide de la pertinence de la réponse par rapport à l'objectif.

8.4.2. Mise à jour on-line simplifiée

Une infobase est un objet dynamique : chaque lecture, chaque recherche enrichit l'utilisateur, qui voudra garder une trace de sa réflexion. Tout comme un document sur papier peut être annoté ou surligné, une infobase peut être enrichie de folios, de groupes et de liens.

Une option de menu permet de passer rapidement en modification. L'utilisateur est alors libre (dans les limites des droits d'accès qui lui ont été conférés) d'insérer, de modifier et de supprimer tout ce qu'il veut.

---

1 Cf. figure 6.2.

2 L'option "Focus" permet à l'utilisateur de ne visualiser dans les folios résultant d'une recherche que les mots de l'expression de recherche entourés d'un contexte limité (quelques mots). La mise en vidéo inverse des termes de recherche au sein des folios de la réponse a aussi pour conséquence de réduire la distance sémantique.

8.4.3. Possibilité de maintenir quelques attributs de formattage

Lorsque nous avons parlé de la création d'infobases, nous avons dit que les balises du texte devaient soit être enlevées, soit être modifiées. Ceci supposait que Views reconnaît quelques attributs de formattage

De fait, il est possible de conserver les codes :

- de gras,
- de souligné,
- de centrage,
- d'alignement à gauche,
- d'alignement à droite, et
- d'indentation.

Comme l'interface de Views est (actuellement) une interface textuelle, il n'est pas possible de visualiser les caractères gras ou soulignés. En pratique, ils sont affichés dans des couleurs ou intensités différentes. Néanmoins, la possibilité de conserver quelques repères visuels dans le texte est très utile au lecteur.

Views est quand même loin de fournir la structure physique d'ODA dont nous avons parlé. Ses codes de formattage sont entièrement disjoints de la structure logique du texte; il n'est pas possible, par exemple, de contraindre l'utilisateur à faire tous ses ajouts de folios dans un style particulier (en souligné, par exemple)[1].

8.4.4. Compression des données

L'infobase, nous l'avons dit, est un fichier contenant toute l'information sur le document : son contenu, son index, sa structure, etc. Le fichier a une particularité supplémentaire : il est comprimé.

Views utilise la "technologie Underhead", marque déposée de Folio Corporation, pour réduire la taille des infobases. Aucune précision n'est donnée dans les manuels à propos de la méthode de compression utilisée, et nous ne pouvons illustrer la théorie qu'en faisant part des taux de compression effectivement observés.

---

1 Du moins n'avons-nous pas trouvé le moyen d'y parvenir ...

Taille de l'original (bytes)	Taille de l'infobase (bytes)	Facteur de compression
50.318	90.112	179 %
131.452	172.032	131 %
545.297	444.416	81 %
2.726.486	1.277.952	47 %
13.632.426	5.480.448	40 %

Figure 8.2. Facteurs de compression observés avec Views

La figure 8.2. donne un petit aperçu des taux de compression atteints par Views. Il faut garder à l'esprit que le fichier final (infobase) inclut non seulement le texte du fichier original, mais aussi l'index, les groupes et les liens.

8.4.5. Sécurité et protection des données

L'utilisation d'une technique de compression a pour conséquence l'abandon des codes ASCII. Cela veut dire que sans Views, une infobase est indéchiffrable et donc inutilisable. La sécurité du système s'en trouve un peu renforcée[1].

D'autre part, au sein même de Views, il est possible de protéger les infobases au moyen de mots de passe. Neuf mots de passe, correspondant à neuf utilisateurs ou classes d'utilisateurs, peuvent être définis. A chaque groupe d'utilisateurs sera assignée une combinaison des droits suivants :

- lire, c'est-à-dire naviguer librement dans l'infobase, à l'aide des liens et des requêtes, sans rien modifier.
- extraire, c'est-à-dire sortir de l'information vers l'extérieur (imprimante ou fichier).
- grouper, c'est-à-dire créer, modifier et supprimer des groupes.
- lier, c'est-à-dire créer, modifier et supprimer des liens.
- modifier, c'est-à-dire passer en mise à jour de l'infobase, avec la possibilité de changer le contenu textuel.
- ajouter, c'est-à-dire créer de nouveaux folios.

---

1 Il y a toujours moyen d'extraire l'information (avec Views) pour impression ... dans un fichier.

8.4.6. Documentation

Pour l'utilisateur comme pour le concepteur d'infobases, une grosse partie de l'apprentissage se fait dans les manuels. Folio propose, avec la version actuelle de Views, deux manuels traduits en français; l'un est destiné à l'utilisateur, l'autre à l'auteur (au concepteur).

La documentation est particulièrement soignée dans sa présentation, claire et agréable à lire[1]. Un petit regret, cependant : les mêmes choses sont répétées à plusieurs reprises, et on a parfois l'impression que le manuel fait partie de la publicité du produit.

Des exercices sont proposés, qui permettent de faire un tour complet des possibilités de Views. D'autre part, une version informatisée du guide de l'utilisateur est disponible. Elle permet la consultation immédiate, au sein de Views, avec toutes les possibilités de recherche et de navigation et est très pratique lorsqu'on est en train d'utiliser le logiciel.

---

1 Malgré les fautes de français ...

Chapitre 9.
Présentation de Spirit

La société ASCII[1] distribue le second produit que nous allons étudier, Spirit. L'acronyme SPIRIT est construit à partir de : système Syntaxique et Probabiliste d'Indexation et de Recherche d'Information Textuelle. Toutes les informations dont nous disposons au sujet de Spirit nous ont été aimablement communiquées par ASCII.

Comme nous l'avons fait pour Folio Views, nous allons, dans un premier temps, expliquer les techniques mises en œuvre au cœur du système, puis, par après, montrer comment elles sont utilisées pour donner à l'utilisateur un confort optimal. Nous aborderons donc d'emblée l'analyseur morpho-syntaxique; nous verrons ensuite les différents types de champs; nous terminerons avec les spécificités que le système Spirit offre à ses utilisateurs : requêtes en langage libre, analyse linguistique, grilles d'interrogation et de visualisation, classement des documents de la réponse, et évolution vers l'analyse sémantique.

9.1. Le processus d'analyse morpho-syntaxique de Spirit

Avant tout, examinons la figure 9.1., sur laquelle est schématisé le fonctionnement de Spirit.

---

1 ASsistance et Conseil en Ingénierie Informatique. La société est située avenue Giraud, 24, à 1030 - Bruxelles.

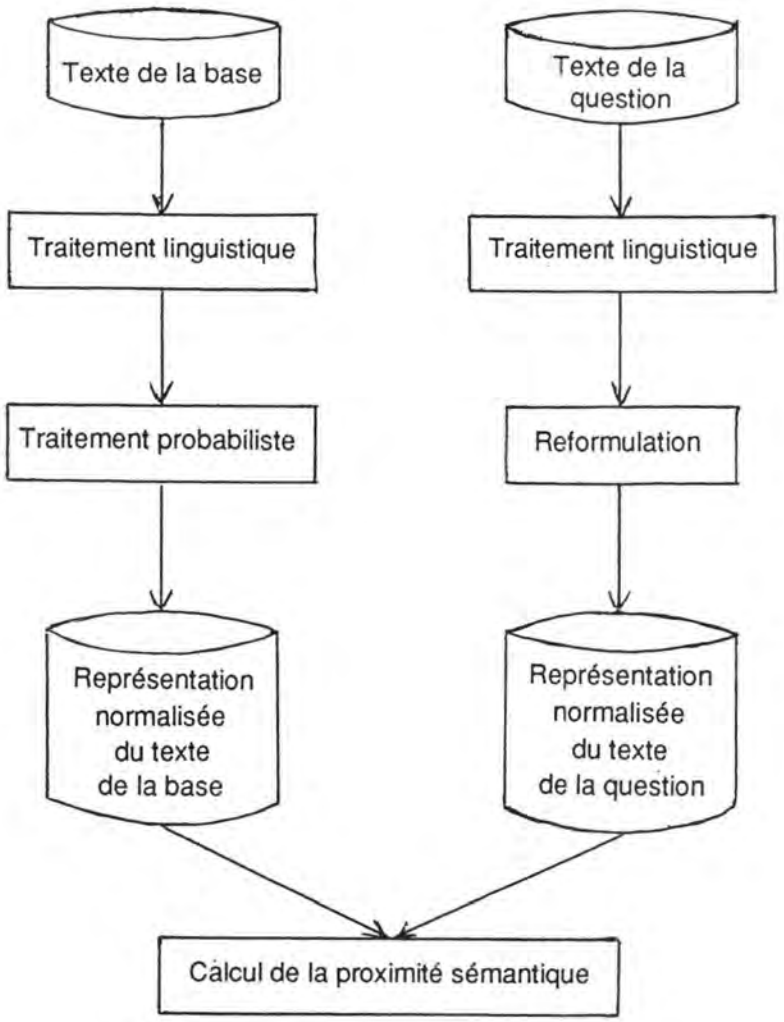

Figure 9.1. Le fonctionnement global de Spirit

On voit tout de suite que ce schéma trouve son origine dans le modèle général des logiciels de recherche documentaire (l'"information retrieval" dont nous avons parlé dans la typologie de l'introduction). On a, d'un côté, l'ensemble des documents[1] et, de l'autre, l'ensemble des requêtes. Le but est de créer une fonction de similitude entre les deux ensembles ou, plutôt, entre les représentations des éléments des deux ensembles.

Pour que la fonction de similitude soit correctement définie, il faut qu'elle ne compare que des choses comparables. C'est le but du traitement linguistique : ramener un texte à un ensemble de mots normalisés, le texte étant dans un cas le document et dans l'autre l'énoncé de la question. Nous

---

1 Etant donné l'orientation "recherche documentaire" de Spirit, nous utiliserons dans ce chapitre le mot "document" à la place de "unité d'information".

verrons que ces mots normalisés appartiennent chacun à une classe grammaticale, ce qui permet de faire la différence, par exemple, entre "the book" (le livre) et "to book" (réserver).

Afin d'optimaliser la fonction de similitude, deux traitements spécifiques sont intégrés à Spirit. Pour les documents, une analyse probabiliste des textes sert à répertorier le nombre d'occurrences de chaque mot normalisé au sein de la base. Les documents des réponses seront classés par ordre de pertinence à partir de ces chiffres : moins un terme d'une question apparaît dans la base, plus il est discriminant, plus son poids, c'est-à-dire son importance, sa "pertinence" a priori, sera élevé. Ce comptage des occurrences des mots normalisés se fait lors de l'introduction des documents dans la base. Il est indépendant des questions posées, et n'est pas appliqué aux questions elles-mêmes.

A l'opposé, la reformulation est un traitement qui n'est effectué que sur les questions. A l'aide d'un thésaurus préalablement construit (on peut évidemment le mettre à jour à n'importe quel moment), les questions seront reformulées : les synonymes, les termes génériques et spécifiques, et les termes associés enrichiront la question pour élargir la réponse.

Comme nous venons de l'évoquer, plusieurs types de relations (autant qu'on veut) peuvent être définis entre deux termes d'un thésaurus de Spirit. Le concepteur définit de manière aussi précise que possible ces relations lors de la création du thésaurus. Il incluera généralement les relations classiques de synonymie (orque "est synonyme de" épaulard)[1] et de hiérarchie ascendante (poirier "est une espèce de" arbre fruitier) et descendante (Asie "est constitué de" Japon, Chine, Corée, etc.). Nous ne parlerons pas de la reformulation et des thésaurus d'une manière plus détaillée, étant donné que ce module logiciel est en option et n'a pas été utilisé dans le cadre de notre petite expérience.

La similitude est calculée, au sein de Spirit, à partir de trois paramètres : le nombre de mots communs (y compris les éventuels mots découlant de la reformulation), les poids de ces mots (au sens probabiliste)[2], et les liaisons syntagmatiques communes. Nous avons déjà vu comment on obtient le poids des mots, il nous reste donc à examiner le processus qui isole les

---

1 La relation de synonymie pose toujours problème : les vrais synonymes sont rares, il suffit de feuilleter le Petit Robert pour s'en apercevoir.

2 Lorsqu'un mot apparaît dans la question suite à la reformulation, son poids est recalculé : le terme original sera toujours préféré aux termes connexes.

mots et identifie les relations syntagmatiques qui les unissent : **l'analyseur morpho-syntaxique.**

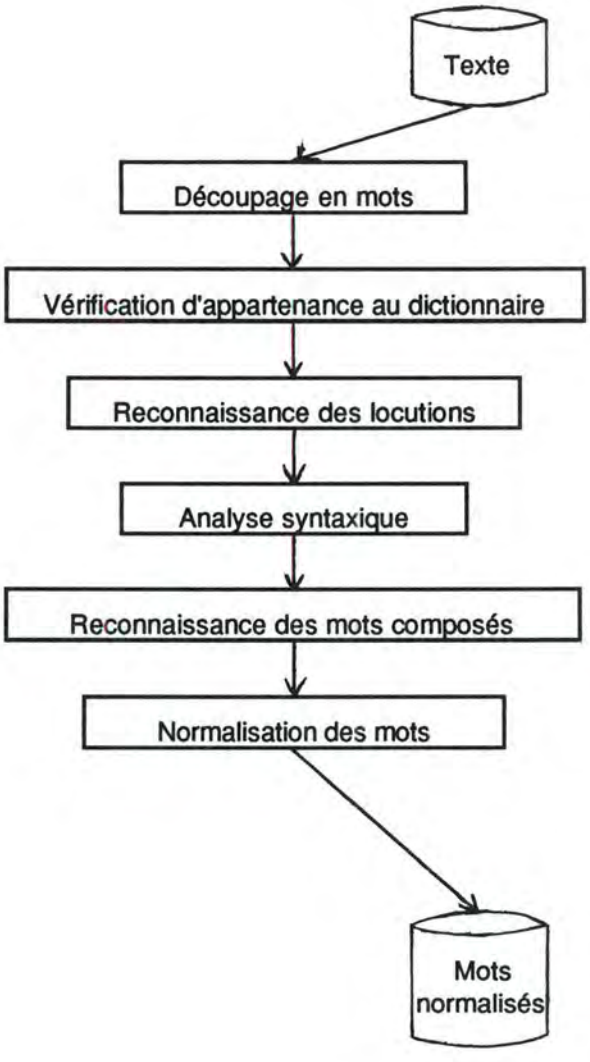

Figure 9.2. Les phases de l'analyseur morpho-syntaxique de Spirit

Comme le montre la figure 9.2., l'analyse morpho-syntaxique de Spirit se décompose en phases bien distinctes et consécutives. Nous allons les passer en revue l'une après l'autre.

9.1.1. Découpage en mots

La première chose à faire avec le texte est de le découper en mots. Nous avons vu la différence entre un mot et une forme[1]; ce sont les mots que

---

1 Un mot est une entité syntaxique, alors qu'une forme est définie au niveau lexical.

Spirit extrait du texte, à l'aide du contexte, de règles linguistiques et de son acquis.

Rappelons que le problème principal est l'ambiguïté des séparateurs. Nous avons eu, lors de l'expérience, quelques exemples de traits d'union malencontreusement placés en fin de ligne. "Simple-minded", coupé en deux par le passage à la ligne, n'a pas été recollé par le module d'entrée, et les modules ultérieurs n'ont pas pu le prendre correctement en considération : "minded" a donné un mot, tandis que "simple-" était catalogué comme un mot invalide.

9.1.2. Vérification d'appartenance au dictionnaire

Maintenant que les mots, briques de base, ont été identifiés, il faut vérifier que chacun d'eux figure dans le dictionnaire électronique de la langue. Ce dictionnaire contient non seulement chaque mot mais aussi toutes ses formes dérivées (le pluriel des substantifs, les formes conjuguées des verbes, etc.). C'est à ce niveau-ci que les textes sont transposés en typographie riche : certains textes sont écrits à l'aide d'un ensemble de caractères limité aux seuls caractères non accentués ou même aux seules majuscules (sans oublier les signes de ponctuation, bien entendu). Les mots mal accentués sont réaccentués, seules les majuscules de début de phrase et celles des noms propres sont maintenues, etc.

Les dictionnaires de Spirit sont limités au langage courant d'une langue, vu que l'ambition des concepteurs est de traiter le langage naturel dans sa généralité. Quand nous avons procédé à l'introduction des textes homéopathiques de Kent dans le système, nous avons constaté que la plupart des mots spécifiques au domaine n'y figuraient pas encore. C'est ainsi que nous avons été amenés à ajouter des mots au dictionnaire anglais ("miasm", "psora", etc.). Pour donner un ordre de grandeur, le dictionnaire anglais comporte 100.000 formes de mots, et est appelé à s'enrichir prochainement de quelques centaines de milliers d'entrées supplémentaires. Le dictionnaire français, quant à lui, comporte 450.000 enregistrements.

9.1.3. Reconnaissance des locutions

Certaines suites de mots ne peuvent pas être dissociées dans le cadre d'une analyse grammaticale. C'est le cas des locutions[1], qui font l'objet d'un traitement particulier au sein de Spirit.

Le critère permettant d'identifier une locution consiste à voir si l'utilisateur voudra trouver le texte contenant la locution quand il n'introduit dans sa question qu'une partie des mots la composant. Dans l'affirmative, le groupe de mots n'est pas une locution. Par exemple, "chemin de fer" est une locution, parce que, a priori, les documents parlant de "chemin de fer" ne seront pas intéressants lorsque la requête contient "chemin" ou "fer" utilisés dans d'autres contextes. Par contre, "recherche documentaire" n'est pas une locution, vu qu'on voudra obtenir les documents qui en parlent quand on cherche sur "gestion documentaire" ou "recherche textuelle". Certains cas sont beaucoup plus difficiles à trancher, à l'instar d'"échiquier politique", qui n'est une locution que pour "échiquier", et pas pour "politique".

Certaines locutions, comme les locutions verbales, peuvent être difficiles à détecter. Prenons un exemple : "avoir l'air". Quand on trouve cette locution sous sa forme conjuguée, il y a déjà une petite difficulté : "il a l'air de s'intéresser à mon travail". La grosse difficulté survient lorsqu'un groupe de mots s'insère à l'intérieur de la locution : "n'ont-ils pas l'air de se désintéresser de nous". Dans des cas pareils, il faudra attendre le résultat de l'analyse syntaxique.

En français, Spirit reconnaît 1500 locutions et expressions idiomatiques. Cela peut paraître peu, mais il ne faut pas perdre de vue que les mots composés tels que "après-midi" et "essuie-mains" (à une ou deux mains, d'ailleurs) ne seront traités que par la suite. Le traitement est différé en raison de la nécessité de disposer des résultats de l'analyse syntaxique qui, elle-même, pour fonctionner correctement, a besoin des locutions en tant que telles et non des mots qui les composent.

9.1.4. Analyse syntaxique

L'analyseur syntaxique est le véritable cœur du système Spirit. Ce module reçoit en entrée des mots ou groupes de mots (locutions) et donne en

---

1 C'est aussi le cas des mots composés, traités ultérieurement. On distingue habituellement ces deux types de compositions de mots par le séparateur utilisé : dans les locutions, les éléments sont séparés par des espaces, alors que les mots composés sont séparés par des traits d'union.

sortie ces mêmes mots, après avoir identifié la ou les classes grammaticales auxquelles ils peuvent appartenir, ainsi que les liens syntagmatiques les soudant les uns aux autres.

Les **classes grammaticales** sont définies par le rôle que peuvent jouer leurs occurrences au sein des phrases. On a, par exemple, la classe des substantifs, celle des verbes, ou encore celle des conjonctions de coordination; Spirit en dénombre une septantaine en français.

La classe grammaticale d'un mot (figurant dans un texte) est identifiée à partir des classes auxquelles ce mot peut appartenir a priori, et aussi à partir du contexte de l'occurrence du mot, qui exclut telle ou telle possibilité[1]. Nous n'aurons jamais, en français du moins, deux articles consécutifs au sein d'un texte; c'est ce qui permet de dire que dans la suite de mots "le la", "la" n'est sûrement pas un article, mais plutôt un substantif.

On comprend à l'aide de cet exemple l'énorme avantage procuré par l'analyse syntaxique : elle permet de lever la plupart des ambiguïtés du langage, et ce, tout à fait automatiquement. Evidemment, toutes les ambiguïtés ne seront pas levées : il en existe qui sont impossibles à lever. Il n'y a qu'à considérer la phrase "la petite brise la glace" pour s'en persuader ! Quand Spirit ne parvient pas à lever une ambiguïté, il mémorise toutes les possibilités restantes. Lors de l'interrogation, un document sera pris en considération s'il comporte un mot de même souche que le mot de la question ayant au moins une classe grammaticale compatible.

Au fur et à mesure que les mots se rangent dans leur classe grammaticale, des **syntagmes**, des groupes de mots formant une unité syntaxique, se dégagent. Les syntagmes sont intéressants parce qu'ils sont généralement identifiables par l'analyse syntaxique, et surtout parce qu'ils véhiculent le sens de l'énoncé. En effet, dans une langue prise de manière globale[2], ce ne sont pas les mots qui sont significatifs (ils sont trop vagues et ambigus), ni les phrases (elles comportent le plus souvent plusieurs idées), mais bien les syntagmes.

---

1 Nous verrons un peu plus loin que c'est grâce à la matrice de proximité des termes que ce processus peut s'exécuter.

2 Dans les sous-langages, ceci n'est plus vrai. Le domaine médical, entre autres, a vu se développer son jargon; la phrase y est réduite à sa plus simple expression, et c'est à son niveau que le sens peut être dégagé, plutôt qu'au niveau du syntagme.

L'analyseur syntaxique a évidemment besoin d'une "connaissance" de la langue. Cette connaissance est concentrée dans deux grandes matrices de proximité des termes, élaborées de manière incrémentale : initialisées à l'aide de l'analyse manuelle d'un texte de quelque 5000 mots, elles s'améliorent ensuite elles-mêmes au fil des textes introduits dans Spirit. En fait, il y a un seuil au-delà duquel les documents introduits n'ont plus d'influence significative sur les matrices. On peut alors s'en servir et les installer chez l'utilisateur. Une matrice est stable dans le temps; cependant, elle variera selon le contexte : les textes techniques ne font pas appel aux mêmes constructions de phrase que les textes journalistiques ou littéraires[1].

9.1.5. Reconnaissance des mots composés

Nous avons déjà vu que Spirit réserve aux locutions une phase de son analyse. En fait, les locutions seront prises par la suite comme un seul mot. Cette simplification n'est pas possible avec l'autre forme de composition de mots, les mots composés. Nous envisageons ici les expressions telles que "grand-mère", "omnisports" et "sous-estimer" qui, selon les cas, seront pris comme une entité ou comme la juxtaposition de leurs composantes.

9.1.6. Normalisation des mots

Dernière étape de l'analyse morpho-syntaxique de Spirit, la normalisation des mots vise à ramener les occurrences des classes grammaticales à leur forme primitive. C'est aussi à ce niveau que seront mis de côté tous les mots dont on ne désire pas garder la trace. Plutôt que de définir une liste noire des formes ou mots indésirables, l'utilisateur de Spirit désignera des classes grammaticales qu'il estime vides de sens[2]. C'est ainsi qu'on écartera le plus souvent les conjonctions de coordination et de subordination, les articles, etc. Ici encore, quelques raffinements sont permis par l'analyse syntaxique : "or" sera ou non écarté selon qu'il fait office de conjonction ou de substantif. On peut raisonnablement espérer filtrer la moitié des mots par ce procédé sans perdre le contenu sémantique du texte. Sur les "Lectures" de Kent, nous avons constaté que seuls 40 à 45 % des mots étaient effectivement indexés.

---

1 Construire les matrices est un processus très long et coûteux, et il est rarissime qu'un utilisateur demande des matrices "sur mesure". C'est utile pour le traitement des sous-langages, par exemple.

2 Spirit permet aussi de définir une liste de mots vides, afin de traiter les cas particuliers.

Les verbes seront donc ramenés à leur forme infinitive, les adjectifs à leur masculin singulier, etc., ce qui n'empêche pas Spirit de conserver aussi la forme grammaticale sous laquelle le mot apparaissait à l'origine.

Nous avons maintenant obtenu ce que nous désirions, à savoir un ensemble de mots normalisés représentatif du document, sur lequel la fonction de similitude peut être définie à la fois simplement et de manière homogène. Le caractère automatique du processus ne rend pas ce dernier parfait; seulement, il acquiert l'avantage de donner des résultats cohérents, ce qui permet de comparer des choses comparables.

9.2. Les types de champs dans Spirit

Au sein de Spirit, tous les morceaux de texte, tous les documents ont la même importance, et sont absolument indépendants les uns des autres. Cela veut dire, en d'autres mots, qu'aucun ordre n'est défini parmi les documents, ils ne forment pas une suite logique comme c'était le cas pour Folio. Pour introduire au sein du système les clés de lecture traditionnelles, il faut définir des informations annexes, qui accompagnent chaque document : ce sont les champs, qui peuvent être textuels, factuels ou non inversés.

Quand nous avons réfléchi à la conception de notre base de textes reprenant le livre de Kent, nous avons pensé que, outre le texte de chaque paragraphe, il fallait enregistrer les informations suivantes :

- le titre de la "lecture"[1] où figure le paragraphe. Il s'agit en fait de "LECTURE" suivi du numéro d'ordre (exemple : "LECTURE I").
- le sous-titre de la lecture. Ce sous-titre peut éventuellement être un passage de l'Organon (exemple : "Par. 1. "The Sick"").
- le type de texte. Nous spécifions ici si le paragraphe est un commentaire (noté "COMM") ou un passage de l'Organon (noté "ORGA").

---

1 Le mot anglais "lecture" signifie "conférence"; au risque de paraître anglophile, nous emploierons indifféremment les deux.

- le numéro de la page. Pour information, nous avons décidé de laisser une trace du support initial et d'enregistrer le numéro de la page sur laquelle débute le paragraphe (exemple : "9").
- le numéro du passage de l'Organon qui est commenté. Si le paragraphe est un commentaire (type = "COMM"), on donne ici le ou les numéros des paragraphes de l'Organon qui sont commentés (exemple : "1 2"). Si le paragraphe est une citation de l'Organon (type = "ORGA") ou s'il ne fait le commentaire d'aucun paragraphe de l'Organon, ce champ est laissé vide.
- le numéro de passage de l'Organon qui est cité. Si le paragraphe est une citation de l'Organon (type = "ORGA"), ce champ indique le numéro du paragraphe dans l'Organon (exemple : "1"). Dans le cas contraire, il est laissé vide.
- le texte du passage de l'Organon qui est commenté. S'il est cité dans le livre de Kent, on reprend ici le texte référencé par le champ donnant le numéro du passage de l'Organon qui est commenté (exemple : "Par. 1. "The physicians's high and only mission is to restore the sick to health, to cure as it is termed.""). Au cas où ce champ est vide ou s'il fait référence à un passage cité par Kent, ce champ-ci reste vide. Ce champ est évidemment redondant et inutile pour la recherche, mais il implémente un moyen d'accès supplémentaire à l'information : on peut voir directement le texte de l'Organon associé au passage du commentaire que l'on a devant les yeux.

Spirit requiert en premier champ un identifiant du document. Nous avons décidé de former l'identifiant à partir de la concaténation des informations suivantes : le titre de la lecture, le type du paragraphe, le numéro de la page et le numéro d'ordre du paragraphe sur la page. Ces deux dernières valeurs suffisaient à rendre l'identifiant unique; nous avons ajouté les autres afin de le rendre significatif.

C'est à la création de la base que l'on choisit la langue et que les champs sont définis; ils acquièrent alors un numéro d'ordre. La mise à jour de la base (ajouts, suppressions et modifications de documents) se fait via le fichier d'entrée, qui contient la suite des documents; chaque document contient la suite ordonnée de ses champs. Chaque champ est précédé d'une ligne sur laquelle figure "$$" suivi du numéro d'ordre du champ. Un extrait de fichier d'entrée est donné à la figure 9.3.

```
$$1
LECT_I_CO_9_1
```

$$2
LECTURE I
$$3
Par. 1. "THE SICK."
$$4
COMM
$$5
9
$$6
1
$$8

Homoeopathy asserts that there are principles which govern the practice of medicine. It may be said that, up till the time of Hahnemann, no principles of medicine were recognized. And even at this day in the writings and actions of the Old School there is a complete acknowledgement that no principles exist. The Old School declares that the practice of medicine depends entirely upon experience, upon what can be found out by giving medicines to the sick. Their shifting methods and theories, and rapid discoveries and abandonment of the same, fully attest the sincerity of their acknowledgment and declarations. Homoeopathy leaves Allopathy at this point, and so in this manner the great division between the two schools is affected. That there are principles Homoeopathy affirms. The Old School denies the existence of principles and with apparent reason, looking at the matter from the standpoint of their practice and methods. They deal only with ultimates, they observe only results of disease, and either deny or have no knowledge of the real nature of man, what he is, where he came from, what his quality is in sickness or in health. They say nothing about the man except in connection with his tissues; they characterize the changes in the tissues as the disease and all there is of the disease, its beginning and its end. In effect they proclaim disease to be something that exists without a cause. They accept nothing but what can be felt with the fingers and seen with the eyes or otherwise observed through the senses, aided by improved instruments. The finger is aided by the microscope to an elongated point, and the microscopic pathological results of disease are noted and considered to be the beginning and the ending, i. e., results without anything prior to them. That is a summary of allopathic teaching as to the nature of sickness. But Homoeopathy perceives that there is something prior to these results. Every science teaches, and every investigation of a scientific character proves that everything which exists does so because of something prior to it. Only in this way can we trace cause and effect in a series from beginning to end and back again from the end to the beginning. By this means we arrive at a state in which we do not assume, but in which we know.
$$9
Par. 1. "The physician's high and only mission is to restore the sick to health, to cure as it is termed".
$$1
LECT_I_CO_10_1
$$2
LECTURE I
$$3
Par. 1. "THE SICK."
$$4
ORGA
$$5
10
$$7
1
$$8

Par. 1. "The physician's high and only mission is to restore the sick to health, to cure as it is termed".
$$1
LECT_I_CO_10_2
$$2
LECTURE I
$$3
Par. 1. "THE SICK."
$$4
COMM
$$5
10
$$6
1
$$8

No controversy will arise from a superficial reading of this statement,

```
and until Hahnemann's hidden meaning of the word "sick" is fully brought to
view, the physician of any school will assent.  The idea that one person will
entertain as to the meaning of the word "sick" will be different at times from
that which another will entertain.  So long as it remains a matter of opinion,
there will be differences of opinion, therefore, the homoeopath must abandon
the mere expressions of opinion.  Allopathy rests on individual opinion and
allopaths say that the science of medicine is based on the consensus of
opinion, but that is an unworthy and unstable foundation for the science of
curing the sick.  It will never be possible to establish a rational system of
therapeutics until we reason from facts as they are and not as they sometimes
appear.  Facts as they appear are expressed in the opinion of men, but facts
as they are, are facts and truths from which doctrines are evolved and
formulated which will interpret or unlock the kingdoms of nature in the realm
of sickness or health.  Therefore, beware of the opinion of men in science.
Hahnemann has given us principles which we can study and advance upon.  It is
law that governs the world and not matters of opinion or hypotheses.  We must
begin by having a respect for law, for we have no starting point unless we
base our propositions on law.  So long as we recognize men's statements we are
in a state of change, for men and hypotheses change.  Let us acknowledge
authority.
$$9
Par. 1. "The physician's high and only mission is to restore the sick to
health, to cure as it is termed".
```

Figure 9.3. Extrait de fichier d'entrée pour Spirit.

Revenons un petit moment à la définition des champs. Chaque champ a un nom, un numéro d'ordre et un code mnémonique. De plus, on lui donne un type; nous aurons des champs textuels, factuels ou non inversés. Voyons quelles sont les caractéristiques et utilisation de chaque type.

9.2.1. Les champs textuels

Comme leur nom l'indique, les champs définis comme textuels sont destinés à contenir les informations textuelles des documents. Techniquement, ces champs se distinguent des autres par le fait qu'eux seuls sont sujets à l'analyse morpho-syntaxique. Lors de la conception d'une base documentaire, il faudra déterminer si un champ peut ou non bénéficier de cette analyse.

Pour les champs textuels, il faut aussi spécifier s'ils sont fournis en typographie riche ou pauvre; dans le deuxième cas, Spirit procèdera à la réaccentuation des mots.

Dans notre exemple, le contenu de chaque paragraphe est stocké dans un champ de type textuel. Nous avons décidé de donner aussi aux champs titre et sous-titre le type textuel, puisque des syntagmes significatifs peuvent y apparaître. Par contre, le neuvième champ, qui contient le texte du passage Organon associé, n'est pas pris comme du texte : aucune recherche ne se fera sur ce champ, et il est inutile de gaspiller de la place et du temps pour la construction d'un index.

9.2.2. Les champs factuels

Les champs factuels contiennent de l'information non textuelle[1] sujette à interrogation. Ils serviront le plus souvent à situer le document par rapport aux autres, à donner des caractéristiques propres à chaque document, ou encore à établir des liens avec l'extérieur (vers des images, par exemple). Lors de l'interrogation, ils sont utilisés pour sélectionner ("et" booléen) les documents voulus.

Nos champs identifiant (champ 1), type de texte (champ 4), numéro de passage commenté (champ 6) et numéro de passage cité (champ 7) ont été déclarés champs factuels. Remarquez que seul le champ identifiant est obligatoire, tous les autres sont facultatifs[2].

9.2.3. Les champs non inversés

Les champs non inversés sont ceux qui en aucun cas ne serviront pour la recherche; ils sont là pour simple information lors de la visualisation. Spirit stockera leur valeur sans pour autant prendre la peine de les indexer. Dans notre expérience, c'est le cas du numéro de page et du texte du paragraphe Organon associé.

9.3. Spécificités de Spirit offertes à l'utilisateur

9.3.1. Requêtes en langage libre

L'apport essentiel de Spirit pour l'utilisateur est évidemment la possibilité d'exprimer les requêtes en langage naturel, c'est-à-dire dans sa langue. Il n'est plus tenu de faire attention à la forme que prend la question, seuls importent le contenu et les mots qu'il emploie.

Plus précisément, les questions sont formulées en langage libre, c'est-à-dire qu'on ne posera généralement pas la question à l'aide d'une phrase : "homeopathic aggravation" sera préféré à "Give me every paragraph speaking of homeopathic aggravation" ou à "Where does one mention the homeopathic aggravation ?". La raison est simple; dans les deux dernières requêtes, on a introduit des mots n'ayant rien à voir avec les documents cherchés ("paragraph", "to speak", "to mention", etc.). Spirit ne

---

1 Ou plutôt, l'information sur laquelle l'analyse morpho-syntaxique n'a pas de raison d'être.

2 Pour Spirit, tous les champs, exception faite de l'identifiant, sont élémentaires, répétitifs et facultatifs.

travaillant pas sur la sémantique des textes mais bien sur leur syntaxe, il ne comprendra pas quels sont les mots qui appartiennent vraiment à la question[1].

9.3.2. Analyse linguistique

L'analyse linguistique met dans les mains de l'utilisateur un outil redoutablement efficace mais parfois trompeur. Par là, nous entendons que le fait de ne pas devoir se préoccuper des différentes formes de mots (pluriels, conjugaisons, etc...) constitue un avantage inestimable, mais qu'on en attend parfois plus de Spirit ! On voudrait, par exemple, que "repetition" soit pris comme terme de recherche lorsque "to repeat" a été mis dans la question, alors que seule une analyse sémantique peut résoudre ce problème.

9.3.3. Interrogation en prenant un document comme question

Spirit donne un autre moyen de mener la recherche : après avoir trouvé un document pertinent, l'utilisateur note son numéro interne, passe en mode d'interrogation par document, et demande les documents similaires. Spirit prend alors l'entièreté du document initial comme question, et classe les documents de la base par ordre décroissant de pertinence, en précisant le nombre de mots communs[2]. Ce mode de recherche peut se révéler très intéressant pour améliorer une réponse dont le taux de précision était médiocre; on peut aussi, mais c'est moins évident, améliorer de cette manière le taux de rappel.

9.3.4. Grilles d'interrogation et de visualisation

Nous n'avons pas encore vu comment utiliser les champs factuels pour l'interrogation. Ce sera chose faite quand nous aurons expliqué les grilles d'interrogation. Les grilles de visualisation, quant à elles, servent simplement à afficher les documents dans un format taillé sur mesure : le concepteur définit à quels endroits de l'écran les champs des documents de la réponse apparaîtront.

---

1 En tout état de cause, on ne fait qu'augmenter le nombre de documents apparaissant dans la réponse. On n'en retire pas, et l'utilisateur pourra toujours sélectionner parmi les classes celles qui lui paraissent les plus pertinentes.

2 Comme nous l'avons vu, la mesure de similitude ne tient pas compte que du nombre de mots communs; le poids des mots et les relations syntagmatiques sont aussi considérées. C'est pourquoi l'ordre décroissant de pertinence peut différer quelque peu de l'ordre décroissant par nombre de mots communs.

Une grille d'interrogation (toujours réalisée par le concepteur) permet d'intégrer plusieurs critères dans la question. On n'est plus limité aux champs textuels, les champs factuels seront désormais des filtres que l'utilisateur combinera pour ne retenir que les documents qui l'intéressent.

C'est sans doute le seul endroit où la notion de "et" booléen apparaît; en effet, les critères liés aux champs factuels agissent comme des "et". Il n'y a pas moyen, par exemple, de demander que tel critère soit rempli ou, à défaut, tel autre. Spirit fait une première sélection sur les champs factuels puis, parmi les documents restants, établit ses classes à l'aide des requêtes sur les champs textuels.

9.3.5. Classement des documents de la réponse

Une autre force de Spirit est de proposer un classement des documents de la réponse. Nous avons vu que la proximité sémantique entre documents et questions était estimée à l'aide d'une fonction de similitude. Trois valeurs interviennent : le nombre de mots communs, le poids de ces mots, et les liaisons syntagmatiques communes.

Pour chaque ensemble de mots communs, et pour chaque ensemble de liaisons syntagmatiques les associant, Spirit donnera une classe de documents. La pertinence globale de la classe sera alors calculée en tenant compte du poids des mots. Les classes sont disjointes, c'est-à-dire que leurs intersections sont vides.

Prenons un exemple. Si la question est "homeopathic aggravation", Spirit donnera probablement quatre classes de réponses. Les deux premières grouperont les documents contenant les mots "homeopathic" et "aggravation", soit associés au sein du même syntagme (première classe), soit non liés grammaticalement (deuxième classe). Les documents des troisième et quatrième classes ne contiendront qu'un seul des deux mots (la troisième pour le plus rare des deux). Cet exemple est très simple; une question comme "is the patient cured if the old symptoms do not come back" génère plus de dix classes. Quant aux questions prenant un document comme point de départ, on arrive rapidement à la centaine de classes.

9.3.6. Tendances vers l'analyse sémantique

La dernière caractéristique de Spirit que nous avons choisi de mentionner ici[1] est la volonté de se tourner résolument vers l'analyse sémantique. La reformulation à l'aide de thésaurus et la prise en compte de la rareté des mots dans le calcul de la similitude sont des étapes importantes, mais le plus gros reste à faire.

Les derniers résultats de la recherche ont mené à pouvoir construire des graphes de concepts. Il s'agit de termes dont la co-occurrence avec liaison syntagmatique dépasse un certain seuil en deçà duquel les assemblages de mots ne sont pas considérés comme pertinents. Après de longues heures (voire journées) de traitement, on obtient un graphe associant les concepts dont l'indice de co-occurrence est significatif et donc, par hypothèse, qui sont fortement liés sémantiquement. Le but ultime que se sont fixé les développeurs est la construction automatique de thésaurus, utiles à la reformulation.

Une autre voie de recherche est l'interrogation multilingue : le projet EMIR (European Multilingual Information Retrieval) est subventionné par la CEE et vise à permettre l'accès multilingue aux bases documentaires. Un compte-rendu des développements et résultats les plus récents est disponible; il s'agit de [RADW91].

---

1 Nous n'avons évidemment pas pu rendre compte de tous les aspects du produit; ce n'est pas l'objet du travail, Spirit n'étant décrit que pour illustrer les concepts théoriques de toute une catégorie de logiciels.

Chapitre 10.
Comparaison entre Views et Spirit

Dans ce dernier chapitre, nous aurons l'occasion de reprendre tous les éléments des chapitres antérieurs pour les assembler et tirer les conclusions de notre expérience, c'est-à-dire mettre en exergue les points positifs et négatifs des logiciels étudiés.

La facilité de concevoir un document électronique et les besoins matériels nécessaires à l'exploitation sont des préoccupations propres au concepteur ou administrateur du système, tandis que les possibilités de recherche, la pertinence des résultats et la satisfaction perçue lors de l'utilisation font partie des critères auxquels les utilisateurs sont les plus attentifs.

Ces pôles d'intérêt différents ont motivé la découpe du chapitre en deux sections principales, l'une consacrée au point de vue du concepteur, et l'autre au point de vue de l'utilisateur. En fin de chapitre, nous donnons un tableau récapitulatif de la comparaison.

10.1. Le point de vue du concepteur

10.1.1. Richesse du document

Chaque logiciel permet de représenter les documents, mais de manières différentes. La découpe en unités d'information est évidemment possible dans les deux cas. La structure du document peut être intégrée dans Views à l'aide des groupes et dans Spirit à l'aide des champs informationnels. Les liens non hiérarchiques ne sont envisageables que dans Views[1].

10.1.2. Input

Le processus de création est similaire pour les deux logiciels : on prépare un fichier "source" et les traitements à y appliquer pour créer le document électronique. On peut souligner que dans aucun des deux cas la

---

1 On peut imaginer des liens relationnels dans Spirit, mais le produit n'est manifestement pas prévu pour cet usage.

sémantique des données ne peut apparaître par un traitement automatique, c'est pourquoi une partie de la création se fait manuellement[1].

A la différence de Spirit, Views est vendu avec une série d'outils utiles à la création. Un de ces outils, VTRANS, permet la conversion en infobase de tout document stocké dans les formats des traitements de texte les plus répandus.

10.1.3. Taille des données

La taille du fichier contenant le livre de Kent est, à l'origine, de 545 Kb. La version électronique du document prend, pour Views, 444 Kb, soit 81 % de la place originelle (la réduction de l'espace requis est due à la compression efficace du texte).

Spirit est plus gourmand : il requiert plus de 2 Mb, soit près de 400 % du fichier de départ. On peut ajouter que la création de la base documentaire dans Spirit prend beaucoup d'espace disque[2].

10.1.4. Taille du programme

Spirit consulte ses dictionnaires lors de l'analyse morpho-syntaxique; il est donc normal que le programme prenne beaucoup plus de place que Views : 20 Mb pour Spirit, 5 pour Views.

10.1.5. Système d'exploitation

Views a le désavantage d'être conçu uniquement pour PC tournant sous DOS; Spirit a le désavantage d'avoir été conçu pour de gros systèmes : seules les fonctions d'interrogation sont disponibles sous DOS. Les systèmes d'exploitation privilégiés de Spirit sont VAX/VMS, UNIX et OS/2.

10.1.6. Protection des données

Etant donné le type de systèmes auxquels Spirit est destiné, il est compréhensible qu'aucune protection ne soit intégrée au programme. Views, par contre, propose, comme nous l'avons vu, plusieurs classes d'utilisateurs, définies par le concepteur.

---

1 L'identification du passage de l'Organon commenté est une information sémantiquement riche et non dérivable automatiquement.

2 Plus de dix fois la taille du fichier de départ, dans notre cas.

10.2. Le point de vue de l'utilisateur

Avant de commencer à donner le point de vue de l'utilisateur, nous aimerions souligner que cette section est le fruit d'une **petite** expérience sur un **petit** document. **Un seul utilisateur**[1] a procédé à la comparaison des deux produits; nous avons tenu compte de son avis dans la mesure du possible, bien qu'il ne nous ait fait part de ses commentaires que fort tardivement.

Pratiquement, cette section trouve sa source dans les observations consignées lors du déroulement de l'expérience, et dans les réponses au questionnaire d'évaluation reproduit en annexe. Nous pensons qu'au delà des résultats de l'expérience, ce sont surtout la démarche et les critères d'évaluation utilisés pour comparer les deux logiciels qui sont importants.

10.2.1. Interface homme / machine

Spirit ne permet que les touches de fonction pour accéder aux différentes parties du programme. En revanche, Views base son interface sur la souris, les menus déroulants et l'environnement multi-fenêtré; les raccourcis clavier sont aussi disponibles pour les utilisateurs fréquents.

Notre utilisateur a jugé que l'effort requis pour manipuler ces deux logiciels est similaire et assez important, mais qu'aucun des deux ne nécessite de longs temps d'apprentissage. Il regrette des options incompréhensibles dans les menus de Views et la laideur (sic) des écrans de Spirit.

Connaissant l'environnement Windows, il espère pouvoir disposer d'un logiciel s'inscrivant dans cette lignée; les boîtes de dialogue, entre autres, lui seraient utiles pour s'orienter dans les parties plus compliquées des programmes.

10.2.2. Formulation des questions

Views offre une panoplie d'opérateurs booléens et positionnels pour la formulation des requêtes. Spirit propose le langage naturel couplé à des filtres booléens sur les champs factuels.

Views, comme Spirit, permet de construire une question à partir du texte d'une unité d'information. Cependant, Views lie les mots sélectionnés

---

1 Notons aussi que cet utilisateur n'a pas le profil de l'utilisateur potentiel : il connaissait l'un des deux logiciels (Views), et est en contact avec l'informatique depuis de nombreuses années.

par des "et" booléens, ce qui a pour effet d'appauvrir la réponse à l'extrême pour peu que le bloc de texte dépasse les quatre ou cinq mots. Enfin, Views a deux spécificités particulièrement utiles : d'une part, il est possible d'enregistrer la question posée en créant un lien requête[1], et, d'autre part, l'index apparaît à l'écran lors de toute interrogation.

Dans le cadre du questionnaire, une des réponses les plus inattendues nous a appris que, sans le thésaurus, Spirit demande plus d'attention que Views lors de la formulation des requêtes. On peut expliquer cette réaction de plusieurs façons :

- l'utilisateur connaît RADAR, qui ne dispose que du "et" booléen pour la recherche.
- la visualisation de l'index est un support à la recherche très apprécié.
- au départ de l'expérience, Views, contrairement à Spirit, n'était pas étranger à l'utilisateur.
- l'expérience s'est d'abord déroulée sous Views, puis sous Spirit.

Il est frappant de constater que l'apparente facilité procurée par les requêtes booléennes cache un gros problème : à aucun moment notre utilisateur n'a varié les opérateurs. En effet, il s'en est toujours tenu à l'espace, c'est-à-dire au "et" booléen, sans utiliser les caractères jokers. Cette attitude a mené à des résultats catastrophiques[2], puisque le taux de rappel chute jusqu'à 0,13 et atteint même 0.

La question par unité d'information dans Spirit a donné de **très bons** résultats. Nous nous sommes rendu compte à cette occasion que la fonction probabiliste de similitude fonctionne à merveille.

10.2.3. Evaluation du résultat

Chacun des systèmes propose une aide à l'évaluation : Views donne l'arbre de recherche (construit au fur et à mesure que la question est formulée), et Spirit classe les unités d'information par ordre de pertinence supposée. Tous deux affichent les passages avec les mots de la requête en mode vidéo inverse; Views dispose de l'option "focus" permettant de limiter

---

1 La création de liens est subordonnée à un droit d'accès.

2 Nous verrons que, paradoxalement, ces résultats ne sont pas perçus comme étant catastrophiques.

le nombre des mots du passage affichés à l'écran, tandis que Spirit implémente la notion de page informationnelle[1].

Les classes de la réponse donnée par Spirit donnent certainement à ce produit un avantage sur Viewsen termes de taux de rappel et de précision. En effet, nous avons constaté que seuls les "et" booléens (sans joker) étaient utilisés dans Views; Spirit a été utilisé avec les mêmes termes de recherche, et les classes d'unités d'information introduisent les "ou" nécessaires à une réponse complète. Dans le même temps, le taux de précision est toujours acceptable si on se limite aux premières classes de la réponse.

Ce qui est surprenant, c'est le fait que l'utilisateur n'est pas conscient de la différence de qualité des réponses. Il pense avoir, dans les deux systèmes, beaucoup de réponses pertinentes et relativement peu de réponses non pertinentes. La seule différence qu'il fait se situe au niveau des "surprises" : il indique que Spirit lui a donné plus de réponses inattendues et positives que Views. Il dit ne pas avoir eu de difficultés à obtenir les réponses pertinentes; quant à l'évaluation de la qualité des réponses, elle est aussi aisée dans Spirit que dans Views.

10.2.4. Temps de réponse

Le temps de réponse ne constitue un obstacle pour aucun des deux systèmes; peut-être sur des machines moins puissantes[2] serait-ce un facteur discriminant (en faveur de Views).

10.2.5. Autres critères

Moins essentiels, les trois derniers critères que nous avons retenus, à savoir la possibilité de parcourir le document sans formuler une question, de modifier le texte pendant l'utilisation, et la qualité de la documentation, sont tous favorables à Views. Il semble que l'utilisateur n'accorde que peu d'importance à ces caractéristiques.

10.2.6. Impression générale de l'utilisateur

On peut dire que l'utilisateur est "moyennement content", déçu surtout par l'interface des logiciels. Il regrette que Spirit ne dispose pas

---

1 Une page informationnelle est un passage de l'unité d'information de la taille d'un écran, où figurent les termes de la question; cette notion est utile surtout lorsque les unités d'information sont longues.

2 Spirit et Views ont été testés sur des compatibles PC à processeur Intel 80386SX.

d'un thésaurus standard : il aimerait autant disposer de Views que de Spirit sans thésaurus, alors qu'il donne un prix subjectif plus élevé pour Spirit doté d'un thésaurus.

10.3. Tableau récapitulatif

La figure 10.1. reprend les informations des sections 10.1. et 10.2. sous forme tabulaire. Les résultats et opinions qui y sont donnés valent uniquement pour notre version du document, et pour l'utilisateur qui a procédé à l'évaluation. En aucun cas il ne faut attribuer à ceci un caractère universel. Ceci clôt le dixième et dernier chapitre du mémoire.

Critère	Views	Spirit
Concepteur		
Richesse des documents		
unités d'information	oui	oui
structure	oui	oui
liens	oui	non
Input automatisable	oui	oui
Taille des données	80 %	400 %
Taille du programme	5 Mb	20 Mb
Système d'exploitation requis	DOS	OS/2 Unix VMS
Protection des données	oui	non
Utilisateur		
Interface		
touches de fonction	oui	oui
souris	oui	non
menus déroulants	oui	non
multi-fenêtrage	oui	non
application Windows	non	non
effort requis	assez grand	assez grand
Formulation des questions		
opérateurs booléens	oui	non
opérateurs positionnels	oui	non
langage naturel	non	oui
champs d'information	non	oui
document comme question	oui	oui
enregistrement d'une question	oui	non
visualisation de l'index	oui	non
Evaluation du résultat		
feedback immédiat	oui	non
classement de la réponse	non	oui
surbrillance des termes	oui	oui
réponse "abrégée"	oui	(oui)
facilité d'évaluation	assez grande	assez grande
taux de rappel	bon	très bon
taux de précision	assez bon	bon
Temps de réponse	très bon	très bon
Navigation sans recherche	oui	non
Modification du document	oui	non
Documentation	oui	non
Impression générale	assez mauvaise	assez mauvaise
Prix subjectif	25.000	30.000

Figure 10.1. Tableau récapitulatif des caractéristiques de Views et de Spirit

Conclusion

Il ressort de l'expérience que dans chacun des domaines étudiés, il y a toujours moyen de progresser. Nous avons pu constater que la modélisation des documents textuels n'a pas encore pris sa véritable dimension au sein des systèmes commerciaux, et que, bien que l'analyse lexicale soit aujourd'hui maîtrisée et que des systèmes beaucoup plus évolués comme Spirit permettent de franchir le cap de l'analyse syntaxique, il reste beaucoup à faire pour que l'analyse sémantique globale devienne une réalité.

En matière de compression, nous avons vu que le regain de recherche a amené des algorithmes de plus en plus efficaces et adaptés aux données textuelles. Dans cette voie, les informaticiens devraient parvenir à se rapprocher davantage de la limite théorique calculée par Shannon.

Mais les deux caractéristiques les plus importantes pour l'utilisateur d'un logiciel de recherche d'information textuelle sont certainement la qualité des réponses obtenues et l'interface homme / machine. Les interfaces évoluent rapidement, et il est probable qu'au fur et à mesure de l'expansion des systèmes de recherche textuelle et de leur apparition massive dans les bureaux, un accent tout particulier sera mis sur la conception d'interfaces plus agréables.

Quant à la qualité des réponses, c'est-à-dire leur pertinence, nous avons vu qu'elle dépend de la formulation des questions, et donc des opérateurs dont le système est pourvu. Les opérateurs booléens et positionnels, malgré leurs résultats médiocres, ont dominé le marché jusqu'à présent; tous les projets de recherche visent à les remplacer.

Cependant, lors de notre expérience, il s'est avéré que l'utilisateur aime aussi pouvoir formuler des requêtes booléennes. C'est pour cette raison que nous avons l'impression que le système idéal serait celui où on pourrait choisir son mode d'interrogation. Nous verrions alors des besoins informationnels satisfaits par le biais d'une recherche combinée, par exemple, sur la sémantique (les concepts) et sur la forme (la suite des caractères) du texte.

Bibliographie

[ANDR90] J. André, V. Quint : Structures et modèles de documents. In [BORN90].

[BELL89] T. Bell, I. H. Witten, J. G. Cleary : Modeling for text compression. ACM Computing Surveys, vol. 21, n°4, 1989.

[BLAI90] D. C. Blair : Language and representation in information retrieval. Elsevier, Amsterdam / New York / Oxford, 1990.

[BLOC89] O. Bloch, W. von Wartburg : Dictionnaire étymologique de la langue française (huitième édition). Presses Universitaires de France, Paris, 1989.

[BORN90] C. Bornes (éditeur) : Le document électronique. INRIA, France, 1990.

[CHOM57] N. Chomsky : Syntactic structures. Mouton, The Hague, 1957.

[CHOM65] N. Chomsky : Aspects of the theory of syntax. MIT Press, Cambridge (Massachusetts), 1965.

[CONK87] J. Conklin : Hypertext : an introduction and survey. IEEE Computer, vol. 18, (9), 1987.

[DACH90] R. Dachelet : Etat de l'art de la recherche en informatique documentaire : la représentation des documents et l'accès à l'information. In [BORN90].

[DANI90] M.-C. Daniel-Vatonne : Hypertextes : des principes communs et des variations. TSI, vol. 9, (6), 1990.

[ELLI90] D. Ellis : New horizons in information retrieval. Library Association, London, 1990.

[FOLI91] Folio Views : Guides de l'utilisateur et de l'auteur. Folio Corporation, 1991.

[FREI91] H. P. Frei, P. Schäuble : Determining the effectiveness of retrieval algorithms. Information processing and management, vol. 27, n° 3, 1991.

[GALL78] R. G. Gallager : Variations on a theme by Huffman. IEEE transactions on information theory, vol. 24, n° 6, November 1978.

[GENE87] G. Genette : Seuils. Editions du Seuil, Paris, 1987.

[GOTT75] D. Gottlieb, S. A. Hagerth, P. H. Lehot, H. S. Rabinowitz : A classification of compression methods and their usefulness for a large data processing center. AFIPS Conference proceedings, vol. 44, 1975.

[HUFF52] D. A. Huffman : A method for the construction of minimum redundancy codes. Proc. IRE, vol. 40, 1952.

[HUTC86] F. L. Hutchins, J. D. Hollan, D. A. Norman : Direct manipulation interfaces. In [NORM86a].

[KIMB88] R. E. Kimbrell : Searching for text ? Send an n-Gram !. Byte, mai 1988.

[LANC68] F. W. Lancaster : Information retrieval systems : characteristics, testing and evaluation (second edition). Wiley, New York, 1968.

[LESU90] R. Lesuisse : Notes du cours sur les systèmes d'information du bureau et d'aide à la décision. Institut d'informatique, FUNDP Namur, année académique 1989 / 1990.

[LUHN58] H. P. Luhn : The automatic creation of literature abstracts. IBM journal of research and development, 2, 1958.

[NORM86a] D. A. Norman, S. W. Draper (editors) : User centered system design. Lawrence Erlbaum Associates, Hillsdale (New Jersey) / London, 1986.

[NORM86b] D. A. Norman : Cognitive engineering. In [NORM86a].

[NUSS91] H. Nussbaumer : Téléinformatique III. Presses polytechniques romandes, Lausanne, 1991.

[PAO89] M. L. Pao : Concepts of information retrieval. Libraries unlimited, Englewood Cliffs, 1989.

[RADW91] K. Radwan, F. Foussier, C. Fluhr : Multilingual access to textual databases. In [RIAO91].

[RIAO91] Recherche d'Information Assistée par Ordinateur 91 : Conference proceedings (intelligent text and image handling). Barcelona (España), april 1991.

[ROBE85] P. Robert, A. Rey : Le grand Robert de la langue française (deuxième édition). Le Robert, Paris, 1985.

[SAFF89] W. Saffady : Text storage and retrieval systems. Meckler, Westport and London, 1989.

[SALT83] G. Salton, M. G. McGill : Introduction to modern information retrieval. McGraw-Hill, New York / London / Toronto, 1983.

[SAVO90] J. Savoy : Les sources des hypertextes : une bibliographie commentée. TSI, vol. 9, (6), 1990.

[SCHA90] L. Schamber, M. B. Eisenberg, M. S. Nilan : Re-examination of relevance : toward a dynamic, situational definition. Information processing and management, vol. 26, n° 6, 1990.

[SHAN51] C. E. Shannon : Prediction and entropy of printed english. The Bell system technical journal, January 1951.

[SHNE87] B. Shneiderman : Designing the user interface. Addison-Wesley, Reading / Tokyo / San Juan, 1987.

[SKAB91] W. Skaba : Towards the ultimate text compression method for help systems. In [RIAO91].

[SPAR71] K. Spark Jones : Automatic keyword classification for information retrieval. Butterworths, London, 1971.

[TENO90] C. Tenopir, J. S. Ro : Full text databases. Greenwood Press, New York / Westport / London, 1990.

[TESK82] F. N. Teskey : Principles of text processing. Ellis Horwood, Chichester, 1982.

[VANR79] C. J. Van Rijsbergen : Information retrieval (second edition). Butterworths, London, 1979.

[WAIN86] J. Wainwright, A. Francis : Office automation, organisation and the nature of work. Gower, 1986.

[ZIPF49] H. P. Zipf : Human behaviour and the principle of least effort. Addison Wesley, Cambridge (Massachusetts), 1949.

[ZIV77] J. Ziv, A. Lempel : A universal algorithm for sequential data compression. IEEE transactions on information theory, vol. 23, n°3, 1977.

[ZIV78] J. Ziv, A. Lempel : Compression of individual sequences via variable-rate coding. IEEE transactions on information theory, vol. 24, n°5, 1978.

Annexes

Questionnaire d'évaluation

Remarques :

- Donnez **votre** point de vue. Soyez franc et subjectif. Surtout, n'essayez pas de raisonner pour d'autres. C'est **votre** opinion qui m'intéresse. Après chaque question et en fin de questionnaire, des espaces libres sont prévus pour les éventuels commentaires. N'hésitez pas à les utiliser pour nuancer vos réponses.

- Vous avez généralement le choix entre entre onze cotes, le non absolu correspondant à 0 et le oui absolu à 10. Entourez le chiffre qui vous paraît le plus approprié, et, si nécessaire, précisez votre pensée à l'aide des zones de commentaires.

1. La découpe du livre en paragraphes s'est-elle révélée être une bonne découpe ?

   non 1 2 3 4 5 6 7 8 9 oui

   .....

   .....

2. Auriez-vous envisagé des morceaux de texte plus petits ou plus grands ?

   non oui, plus petits oui, plus grands

   .....

   .....

3. A supposer que vous puissiez disposer gratuitement d'une machine dotée de Folio ou de Spirit, quel produit choisiriez-vous ?

   Folio non 1 2 3 4 5 6 7 8 9 oui
   Spirit non 1 2 3 4 5 6 7 8 9 oui

   .....

   .....

4. Connaissant les exigences matérielles de chacun des logiciels, lequel choisiriez-vous ?

   Folio non 1 2 3 4 5 6 7 8 9 oui
   Spirit non 1 2 3 4 5 6 7 8 9 oui

   .....

   .....

5. Combien seriez-vous prêt à payer pour acquérir chacun des logiciels (indépendamment du prix réellement pratiqué) ?

Folio : .....
Spirit : .....

.....

.....

6. Pensez-vous qu'une longue période de formation est nécessaire pour l'apprentissage de l'utilisation de chacun des logiciels ?

Folio non 1 2 3 4 5 6 7 8 9 oui
Spirit non 1 2 3 4 5 6 7 8 9 oui

.....

.....

7. Trouvez-vous que la formulation des questions demande beaucoup d'attention ?

Folio non 1 2 3 4 5 6 7 8 9 oui
Spirit non 1 2 3 4 5 6 7 8 9 oui

.....

.....

8. Trouvez-vous que l'évaluation de la qualité des réponses est aisée ?

Folio non 1 2 3 4 5 6 7 8 9 oui
Spirit non 1 2 3 4 5 6 7 8 9 oui

.....

.....

9. En général, avez-vous l'impression d'avoir obtenu beaucoup de réponses pertinentes ?

Folio non 1 2 3 4 5 6 7 8 9 oui
Spirit non 1 2 3 4 5 6 7 8 9 oui

.....

.....

10. En général, avez-vous l'impression d'avoir obtenu beaucoup de réponses non pertinentes ?

Folio non 1 2 3 4 5 6 7 8 9 oui
Spirit non 1 2 3 4 5 6 7 8 9 oui

.....

.....

11. Les questions posées donnent-elles souvent des réponses inattendues et positives ?

```
Folio      non  1  2  3  4  5  6  7  8  9  oui
Spirit     non  1  2  3  4  5  6  7  8  9  oui
```

.....

.....

12. Les questions posées donnent-elles souvent des réponses inattendues et négatives ?

```
Folio      non  1  2  3  4  5  6  7  8  9  oui
Spirit     non  1  2  3  4  5  6  7  8  9  oui
```

.....

.....

13. Avez-vous l'impression que les temps de réponse étaient longs ?

```
Folio      non  1  2  3  4  5  6  7  8  9  oui
Spirit     non  1  2  3  4  5  6  7  8  9  oui
```

.....

.....

14. Avez-vous eu des difficultés à obtenir les réponses pertinentes ?

```
Folio      non  1  2  3  4  5  6  7  8  9  oui
Spirit     non  1  2  3  4  5  6  7  8  9  oui
```

.....

.....

15. Hormis la formulation des questions, trouvez-vous que ces logiciels sont faciles à utiliser ?

```
Folio      non  1  2  3  4  5  6  7  8  9  oui
Spirit     non  1  2  3  4  5  6  7  8  9  oui
```

.....

.....

16. D'une manière générale, ces logiciels vous ont-ils semblé agréables à utiliser ?

```
Folio      non  1  2  3  4  5  6  7  8  9  oui
Spirit     non  1  2  3  4  5  6  7  8  9  oui
```

.....

.....

17. A supposer que l'on construise un troisième système, quelles caractéristiques de Folio souhaiteriez-vous lui donner ?

.....

.....

.....

18. A supposer que l'on construise un troisième système, quelles caractéristiques de Spirit souhaiteriez-vous lui donner ?

.....

.....

.....

19. A supposer que l'on construise un troisième système, quelles caractéristiques de Folio souhaiteriez-vous lui éviter ?

.....

.....

.....

20. A supposer que l'on construise un troisième système, quelles caractéristiques de Spirit souhaiteriez-vous lui éviter ?

.....

.....

.....

21. A supposer que l'on construise un troisième système, quelles caractéristiques ne figurant ni dans Folio ni dans Spirit souhaiteriez-vous lui donner ?

.....

.....

.....

22. Veuillez consigner ici tous les commentaires que vous n'avez pas eu l'occasion de formuler plus tôt dans le questionnaire.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Merci pour votre collaboration

BAL_TABL.FSR

```
: enlever les balises de tableau restant dans le fichier presque prêt.
:
"\<DT\>" "" -c
"\<FT\>" "" -c
```

BUILDFSR.FSR

```
: prendre l'ensemble des lignes de la table des matières et créer des règles
:   FSR pour remplacer la version courte de la table des matières par la
:   version longue, incluant les sous-titres.
:
"[-]*☻[-]*{[^\r\n-]+}\r\n☻{[^‡]+}[‡]+\r\n" "\"{1}\254\" \"{1} \\\-♥{2}\" \-c\r
```

CLEAN_TM.FSR

```
: nettoyer la table des matières.
:
: laisser seulement la première ligne avec un code de reconnaissance
"[-]*☻[-]*{[^\r\n-]+}\r\n☻[^‡]+[‡]+\r\n" "\<\{{1}\}\~\254>\r\n" -c
```

CONV1.FSR

```
: certains caractères du VAX sont convertis en leur équivalent DOS ("º" "§").
"\167" "\21" -c
:
: les tags <DO>, <FO>, <DI> et <FI> (passages Organon et italiques) sont
:   convertis en ASCII 249, 250, 251 et 252
"\<DO\>" "\249" -c
"\<FO\>" "\250" -c
"\<DI\>" "\251" -c
"\<FI\>" "\252" -c
:
: les tags <DC> et <FC> (citation dans le texte) sont à bazarder
"\<[DF]C\>" "" -c
:
: les espaces et tabulations de fin de ligne sont à supprimer, ainsi que les
:   espaces de début de ligne.
"[ \t]+\r\n" "\r\n" -c
"\r\n[ ]+" "\r\n" -c
:
: les espaces, tabs et returns ne doivent pas apparaître consécutivement.
" [ ]+" " " -c
"\t[\t]+" "\t" -c
"\r\n(\r\n)+" "\r\n" -c
:
: les numéros de page sont à supprimer.
"\#P [0-9]+\r\n" "" -c
```

CONV2.FSR

```
: mettre en gras les passages de l'organon; les indenter
"\t\249{[^\249\250]+}\250" "♥-{1}‡" -c
:
: mettre en souligné les passages en italique du texte
"\251{[^\251\252]+}\252" "{1}" -c
:
```

```
: supprimer les tabulations restantes (en milieu de ligne).  Il faut tenir
:   compte des espaces avant et après (pas d'espaces consécutifs).
"{[^\r\n ]}[ ]*\t[ ]*" "{1} " -c
:
: gérer les returns de fin de ligne écran et de fin de phrase.
"{[\.\?\!\"\)\>]}\r\n{[^\t♥\#]}" "{1}  {2}" -c
"{[^\.\?\!\"\)\>]}\r\n{[^\t♥\#]}" "{1} {2}" -c
```

REQUOTE.FSR

```
: redonner au caractère ASCII 253 sa valeur initiale de quote
:
"\253" "\"" -c
```

SEGMENT.FSR

```
: segmenter avec les tabulations ou indentations de début de ligne.
"\r\n{[\t♥]}" "\r\n\r\n\►\r\n{1}" -c
:
: gérer les autres # (en faire les seuls folios référencés).
"#R +{[^ \r\n]+} +{[^ \r\n]+}\r\n\#D +{[^ ]}{.+}\r\n" "\r\n\r\n► {1}.{2}\r\n\r
```

UNQUOTE.FSR

```
: remplacer quotes par le caractère ASCII 253
:
"\"" "\253" -c
```

CONV.BAT

```
fsr p filtre.1 /o /f\jean\fsr\conv1.fsr
del p
fsr filtre.1 filtre.2 /o /f\jean\fsr\conv2.fsr
del filtre.1
fsr filtre.2 refer /o /f\jean\fsr\segment.fsr /r
del filtre.2
```

CREA.BAT

```
insert hierarch philknt.ins \jean\fige\balises.tab /o
del 1.ext
del 2.ext
del hierarch
fsr philknt.ins philknt.fin /o /f\jean\fsr\bal_tabl.fsr
del philknt.ins
create philknt.fin table_m.fin philknt.nfo /o /s10 /b"JM pour Archimède" /t"Le
del philknt.fin
del table_m.fin
tpgcopy \jean\fige\philpage.nfo philknt.nfo
```

EXTR.BAT

```
extract \jean\fige\philknt.doc \jean\p \jean\fige\balises.tab /o
del 1.ext
copy \jean\fige\tab175 1.ext
del 2.ext
copy \jean\fige\tab229 2.ext
del extract.dat
copy \jean\fige\philknt.doc \provi
```

PHILKNT.BAT

```
call extr.bat
call conv.bat
call tm.bat
call crea.bat
```

TM.BAT

```
grabtext refer lignes12.raw \jean\fige\centre.cod /l /o
fsr lignes12.raw lignes12.unq /o /f\jean\fsr\unquote.fsr
del lignes12.raw
fsr lignes12.unq lignes1.tm /o /f\jean\fsr\clean_tm.fsr
fsr lignes12.unq make_tm.fsr /o /f\jean\fsr\buildfsr.fsr
del lignes12.unq
hierarch refer hierarch lignes1.tm lignes1.fin /m /p" Where am I ?" /o
del refer
del lignes1.tm
fsr lignes1.fin table_m.unq /o /fmake_tm.fsr
del lignes1.fin
del make_tm.fsr
fsr table_m.unq table_m.fin /o /f\jean\fsr\requote.fsr
del table_m.unq
```